Compressed Post-Quantum Silent OT from Isogenies

Pouria Fallahpour[1], Arthur Herlédan Le Merdy[2⋆], and Mahshid Riahinia[3⋆⋆]

[1] Sorbonne Université, CNRS and LIP6, France
[2] COSIC, KU Leuven, Heverlee, Belgium
[3] DIENS, École normale supérieure, CNRS, Paris, France
pouria.fallahpour@lip6.fr, arthur.herledanlemerdy@esat.kuleuven.be, mahshid.riahinia@ens.fr

Abstract. Oblivious transfer (OT) is a foundational tool in secure computation. Random OT-correlated pairs of the form (r_0, r_1) and (b, r_b), where $r_0, r_1, b \xleftarrow{\$} \{0,1\}$, significantly speed up secure computation protocols. Pseudorandom correlation functions (PCFs), introduced by Boyle et al. (FOCS 2020), are an efficient means of non-interactively generating correlated pairs. In a PCF, two parties store short keys that are used to locally generate large amounts of correlated pairs. While pre-quantum PCFs for OT achieve highly compact key sizes (e.g., 30 kB), a significant gap remains in the post-quantum regime, where existing solutions rely on lattices or learning-parity-with-noise-style assumptions and yield much larger keys.

We construct compact post-quantum PCFs for OT. Our construction has key sizes in the range of 100 kB, approaching the compactness of pre-quantum PCFs and seven times smaller than the most compact state-of-the-art post-quantum alternatives. The key size of our PCF remains invariant regardless of the target number of OTs, in contrast to prior alternatives. We estimate a throughput of 7 OTs per second for our PCF. We additionally provide a security proof in the quantum random oracle model, constituting the first such analysis of a post-quantum PCF. The core of our construction is a new, compact constrained pseudorandom function (CPRF) for inner-product membership predicates. Its security is based on a natural extension of the parallelization problem, which we introduce as the parallelization problem with auxiliary inputs. We provide an extensive analysis of this assumption as well as an implementation of our construction.

⋆ Part of this work was done while Arthur Herlédan Le Merdy was at ENS de Lyon.
⋆⋆ This work was done while Mahshid Riahinia was visiting IRIF, Paris.

Table of Contents

Compressed Post-Quantum Silent OT from Isogenies 1
Pouria Fallahpour, Arthur Herlédan Le Merdy, and Mahshid Riahinia
1 Introduction 3
2 Technical Overview 6
3 Preliminaries 12
3.1 Pseudorandom Functions 12
3.2 Constrained Pseudorandom Functions 13
3.3 Pseudorandom Correlation Functions 15
3.4 Cryptographic Group Actions 17
3.5 Elliptic Curves and Isogenies 19
3.6 Oriented Elliptic Curves 20
3.7 Quantum Preliminaries 20
4 CPRFs from Group Actions 21
4.1 A No-Evaluation Unpredictable CPRF for IPM Predicates 21
4.2 Achieving Adaptive Security 23
4.3 Instantiation of the Group Action 27
4.4 Assumption Analysis 29
5 Putting Everything Together: PCFs for OT from Isogenies 34
5.1 Achieving NIST Level 1 security. 35
5.2 Efficiency Analysis 36
5.3 Key Compression of the CPRF 38
5.4 Implementation 39

1 Introduction

Oblivious Transfer (OT) is a foundational tool in secure computation protocols [51,30,36], where parties intend to evaluate an arbitrary circuit over their private inputs. In a chosen-bit OT protocol, a sender holds two messages (m_0, m_1) and a receiver holds a choice bit $\mathsf{b} \in \{0,1\}$. At the end of the protocol, the receiver should learn m_{b}, without gaining any information about $m_{1-\mathsf{b}}$, and the sender should learn nothing about b. Known constructions of OT protocols rely on computationally expensive public-key operations, and therefore scale poorly for modern secure computation applications that require millions of OTs. To overcome this issue, the idea of the *preprocessing* model has emerged. This paradigm divides the protocols in two phases; an offline, input-independent *preprocessing* phase where parties generate a massive supply of *random* OTs, and a highly efficient *online* phase, where these pairs are consumed to securely and rapidly compute the intended function.[4] A random OT-correlated pair consists of (r_0, r_1) given to the sender, and (b, r_b) given to the receiver, where $r_0, r_1 \xleftarrow{\$} \{0,1\}^\lambda$ are two random strings and $b \xleftarrow{\$} \{0,1\}$ is a random bit. The protocol of [43] shows that if two parties have access to $O(n)$-many random OT-correlated pairs, then they can evaluate any Boolean circuit of n gates over their private inputs extremely fast, i.e., while communicating only 4 bits per AND gate. Consequently, the main bottleneck in the efficiency of secure computation has shifted to the offline phase. So far, two main directions exist for efficiently generating large amounts of OT correlations: OT extensions and silent OT.

OT Extensions. Introduced by [10], this approach assumes that two parties have access to some *base* OTs and processes them to be extended to a large number of OTs. The work of [45] proposes a highly efficient OT extension protocol—at 34 million OTs per second—using only lightweight symmetric-key cryptography. However, this protocol requires parties to communicate a lot to generate every extended OT, i.e., λ bits per OT. More recently, SoftSpokenOT [57] improved this by requiring λ/k bits of communication for a small k (e.g., $k = 5, 8$) at the cost of slightly increasing the computation time.

Silent OT. Silent OT protocols are solutions that are optimized for communication; they generate random OTs without any interactions (hence silently). Each of these random OTs can be used to derive a chosen-bit OT by communicating only 3 bits per OT [9], which is the optimal communication. In turn, silent OT protocols either present less efficiency or require heavier cryptographic tools, e.g., public-key cryptography, or ad-hoc assumptions. For example, the silent OT of [56] is 8 times slower than SoftSpokenOT and assumes the hardness of a new variant of the Learning Parity with Noise (LPN) problem.

[4] Note that the offline phase is independent from the inputs of the parties. Therefore, the parties are limited to generating *random* OTs in this phase. If the secure computation protocol requires chosen-bit OTs, the parties need to interact in the online phase to transform their random OTs to chosen-bit OTs, which is done by communicating only 3 bits per OT.

Pseudorandom Correlation Functions (PCFs) [16] were introduced as a means of silently generating correlated randomness on demand. A PCF enables two parties to store short, correlated keys, k_0 and k_1, and subsequently generate a superpolynomial number of pseudorandom correlated pairs locally, without any interaction. The study of PCFs has recently gained significant attention and multiple constructions that offer different levels of efficiency are known [16,28,15,27,19]. While pre-quantum constructions can achieve key sizes as small as 30 kilobytes [19], there remains a substantial gap in the post-quantum regime. This disparity stems from the underlying assumptions required for post-quantum security; existing constructions rely on either (1) lattice-based assumptions via fully homomorphic encryption (FHE) [16,44], which require key sizes of several gigabytes, or (2) learning parity with noise (LPN)-style assumptions, which often depend on ad-hoc LPN variants [16,27,15], and/or require megabyte-sized keys [18]. To date, the most compact post-quantum construction [16,27] achieves a key size of 370 kilobytes. Furthermore, we note that all known LPN-based PCFs lack optimal asymptotic performance; rather than scaling efficiently, their reported key sizes are tied to a fixed target capacity (e.g., 2^{30} OTs) and increase considerably when instantiated for larger targets. We therefore ask the following question:

Can we build asymptotically-compact post-quantum PCFs for OT?

Our Contribution. In this work, we answer the aforementioned question in the affirmative by introducing the first highly compact PCFs for OT from *isogenies*. Our post-quantum construction yields key sizes in the range of 100 kilobytes, approaching the compactness of pre-quantum PCFs, and remains invariant to any number of target OT outputs. Prior to our work, the most compact state-of-the-art post-quantum PCFs [15] had key sizes of 370 kilobytes for a fix target of 2^{30}, which increases considerably for larger targets, e.g., the key size increases to 650 kB for a target of 2^{48} OTs. We estimate a throughput of 7 OT/s for our PCF on a single core of a standard computer. The security of our PCF reduces to three assumptions:

1. hardness of the parallelization problem with auxiliary inputs (Para-AI). We introduce and analyze this problem, which can be viewed as a natural extension of the well-known parallelization problem [29]. The parallelization problem is the analogue of the computational Diffie-Hellman problem for group actions.
2. existence of correlation-robust hash functions *or* random oracles.
3. security of the weak pseudorandom function (wPRF) of Boneh et al. [13] (henceforth BIPSW).

At the core of our construction lies a new, compact *constrained pseudorandom function* (CPRF) for inner-product membership predicates built from Para-AI. We provide an extensive analysis of Para-AI and argue its credibility. Similarly to multiple previous results on CPRFs [6,19,2], we initially prove the security of our scheme in the no-evaluation setting, where the adversary cannot issue

evaluation queries. This can be lifted to full security in the random oracle model (ROM) using a compiler inspired by [6]. We revisit this transformation and improve it in several aspects; most notably, our modifications allow us to base the security of our CPRF directly on the hardness of a *search* problem. Moreover, we analyze our construction, when instantiated with a random oracle, in the quantum random oracle model (QROM). To the best of our knowledge, this constitutes the first QROM analysis of a post-quantum PCF. Notably, both our ROM transformation for achieving full security and our QROM analysis are broadly applicable to other works that use similar techniques to ours [6,19,2].

Finally, we provide a concrete implementation of our construction. We are optimistic that these results will pave the way for exploring alternative isogeny-based solutions in secure computation contexts. We outline several promising directions for future research. We provide a comparison between our PCF with other post-quantum PCFs in Table 1.

Table 1. Comparison between our post-quantum PCF for OT with existing post-quantum PCF constructions, for a security level of $\lambda = 128$.

Result	Assumption*	Asymptotic	OT/sec	Key-Size (kB)	
				$\lvert k_0\rvert$	$\lvert k_1\rvert$
[16]	ring-LWE (FHE)	✓	✗†	$> 1,000,000$	
[16,27]	VD-LPN	✗	1k‡	≈ 2000	≈ 2000
[15]	EA-LPN	✗	118k‡	370	370
[18]	sparse LPN	✗	507k‡	2812	638
[44]	ring-LWE (FHE)	✓	9	$> 1,000,000$	
Our Work§	Parallelization with Auxiliary Inputs (Def. 17)	✓	7	0.016	103

* All constructions additionally assume the ROM. To the best of our knowledge our work is the first to also provide an analysis in the QROM.
† The construction relies on fully homomorphic encryption and generates a single OT in $O(\lambda^2) \approx 16000$ seconds.
‡ The performance is evaluated for a fixed target of 2^{30} correlations rather than in the asymptotic regime. The numbers are taken from [18, Table 1].
§ Numbers are estimated for generic group actions and with respect to the C implementation of the qt-`PEGASIS` framework presented in [31, Table 1].

2 Technical Overview

In this work, we follow the framework proposed by [19] (henceforth BCMPR) for building PCFs for OT. This framework can be essentially viewed as a compiler from *pseudorandomly-constrained pseudorandom functions* to PCFs for OT. A constrained pseudorandom function (CPRF) is a pseudorandom function $\mathsf{CPRF}_{\mathsf{msk}} : \mathcal{X} \to \mathcal{Y}$ that allows generation of a *constrained key* ck_C for a predicate (called constraint) $C : \mathcal{X} \to \{0,1\}$ such that ck_C can correctly compute the output of $\mathsf{CPRF}_{\mathsf{msk}}$ on all x satisfying $C(x) = 0$. Moreover, ck_C should reveal no information about the output of the function on any other input. A pseudorandomly-constrained PRF [19] is a CPRF that supports a pseudorandom predicate as a constraint. In the following, we recall how such a CPRF is used in the framework of [19] to build PCFs for OT. We later show how we instantiate it from isogenies, and discuss our theoretical contributions and practical optimizations.

From CPRFs to PCFs [19]. Let $F : \mathcal{K} \times \mathcal{X} \to \{0,1\}$ be a family of pseudorandom functions (predicates). Let CPRF be a constrained PRF that supports both F and $\overline{F}$ as constraints, where $\overline{F}$ denotes the complement of F, i.e., $\overline{F}_k(x) := 1 - F_k(x)$ for all $k \in \mathcal{K}, x \in \mathcal{X}$. Given these two ingredients, the BCMPR framework builds a PCF for OT as follows. The sender gets a PCF key that consists of two master secret keys $\mathsf{msk}_0, \mathsf{msk}_1$ of CPRF. The receiver gets, as its PCF key, a random PRF key $k \xleftarrow{\$} \mathcal{K}$ of F, together with two constrained keys $\mathsf{ck}_0, \mathsf{ck}_1$. The first constrained key ck_0 evaluates $\mathsf{CPRF}_{\mathsf{msk}_0}$ on all inputs satisfying $F_k(x) = 0$, and the second constrained key evaluates $\mathsf{CPRF}_{\mathsf{msk}_1}$ on all inputs satisfying $\overline{F}_k(x) = 1$ (i.e., $F_k(x) = 0$). Once the keys are set, the parties can locally evaluate OT pairs as follows: on any input x, the sender outputs (r_0, r_1) where r_b is the evaluation of CPRF on x using msk_b. And the receiver outputs (b, r), where $b = F_k(x)$ and r is the output of ck_b on x. These output pairs are correctly OT correlated, since on any input x, the constrained key ck_b evaluates to the same thing as msk_b on all input satisfying $F_k(x) = b$. Therefore the receiver correctly outputs (b, r_b). Also, since, on any input, the constrained key ck_{1-b} fails in computing the output of msk_{1-b}, the security of the CPRF implies that the receiver, does not learn anything about r_{1-b}. On the other hand, since the PRF key k is hidden from the sender, this party cannot distinguish b from random, thanks to the pseudorandomness of the PRF F.

Therefore, to obtain an efficient PCF for OT through the BCMPR framework, one needs to (1) find a low-complexity PRF F and (2) a CPRF that is strong enough to support F and $\overline{F}$ as constraints. The works of [19,26] instantiate this framework by replacing F by a *weak* PRF (wPRF) (i.e., that is pseudorandom on uniformly random inputs) that can have lower complexity. In turn, the obtained PCF is a *weak* PCF, in which (similarly to PRFs vs wPRFs) the security properties are guaranteed on uniformly random inputs.[5] [19] then

[5] Contrary to wPRFs, weak PCFs are the default notion in the PCF literature, and similarly to wPRFs, they can be easily upgraded to strong PCFs in the random oracle model.

observes that multiple wPRF constructions in the literature can be presented as *inner-product membership* (IPM) predicates, i.e., the wPRF is associated with a set S and $F_k(x) = 0$ *iff* $\langle k, x\rangle \in S$. Examples include the learning with rounding wPRF of [8], that of [13] (henceforth, BIPSW), and the Goldreich-Applebaum-Raykov construction [41,5]. Note that the class of IPM predicates is probably the weakest class in terms of complexity that can contain weak PRFs; an adversary can efficiently learn the key of any keyed function that can be evaluated linearly in its key, after seeing enough outputs. To sum up, the BCMPR framework reduces the problem of building a PCF for OT to building a CPRF that supports inner-product membership predicates as constraints.

CPRFs for IPM Predicates from Isogenies. Our core idea follows the approach of [29,3] and many others in the literature: leverage the common formalism that exists between discrete-log-based problems and isogeny-based problems, which is *group actions*. We reinterpret the group based CPRF of [19] in terms of group actions, then reinstantiate it from isogenies and analyze the resulting construction and security assumption. We first recall the definition of a group action. We say that a group G, denoted multiplicatively, acts on a set T if there is a map $\star : G \times T \to T$ such that for all $t \in T$ and $g_1, g_2 \in G$, it holds that (I) $e_G \star t = t$, where e_G is the identity in G, and (II) $g_1 \star (g_2 \star t) = (g_1 g_2) \star t$. In the following, we additionally require G to be an abelian group and to act regularly, i.e., for any pair of set elements (t_1, t_2), there exists a unique group element g such that $g \star t_1 = t_2$.

Our CPRF is defined over binary inputs $\boldsymbol{x} \in \{0,1\}^n$. The master secret key and its evaluation are computed as follows:

- $\mathsf{CPRF.KeyGen}(1^\lambda)$: Sample $(g_1, \ldots, g_n) \xleftarrow{\$} G^n$. Sample $t \xleftarrow{\$} T$. Output $\mathsf{msk} := (t, (g_i)_{i\in[n]})$.[6]
- $\mathsf{CPRF.Eval}(\mathsf{msk}, \boldsymbol{x})$: Output $\prod_{i=1}^{n} g_i^{x_i} \star t$.

The two algorithms above essentially represent the Naor-Reingold PRF [50] in terms of group actions which is proposed by Boneh, Kogan and Woo [14]. The pseudorandomness of the evaluation algorithm follows assuming the hardness of the group action variant of the decisional Diffie-Hellman (DDH) problem in G. When G is generated by a group element g_0, this assumption requires that the tuple $(g_0^a \star t, g_0^b \star t, g_0^{a+b} \star t)$ is computationally indistinguishable from $(g_0^a \star t, g_0^b \star t, g_0^c \star t)$, where $a, b, c \xleftarrow{\$} [\mathrm{ord}(G)]$ and t is a fixed public set element. We now explain how to extend this construction to a constrained PRF for IPM predicates.

To generate a constrained key for an IPM constraint defined by a vector $\boldsymbol{k} \in \{0,1\}^n$ and a set $S \subseteq [n]$, we do as follows:

- $\mathsf{CPRF.Constrain}(\mathsf{msk}, (S, \boldsymbol{k}))$: Sample $r \xleftarrow{\$} G$. For all $s \in S$ define $t_s := r^s \star t$. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$, where each $\alpha_i := r^{-k_i} \cdot g_i$. Output $\mathsf{ck} := (\boldsymbol{k}, (t_s)_{s\in S}, \boldsymbol{\alpha})$.

[6] We denote by $[n]$ the set $\{1, \ldots, n\}$.

– $\mathsf{CPRF.CEval}(\mathsf{ck}, \boldsymbol{x})$: On input $\boldsymbol{x} \in \{0,1\}^n$, compute $s = \langle \boldsymbol{x}, \boldsymbol{k} \rangle$. If $s \notin S$, output $\perp$. Otherwise, output $\prod_{i=1}^{n} \alpha_i^{x_i} \star t_s$.

The correctness follows easily: replacing each α_i by $r^{-k_i} \cdot g_i$ in the constrained evaluation, it can be re-written as $\mathsf{CEval}(\mathsf{ck}, \boldsymbol{x}) = \prod_{i=1}^{n} g_i^{x_i} \cdot r^{-\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \star t_s$. For any input $\boldsymbol{x}$ satisfying $s = \langle \boldsymbol{x}, \boldsymbol{k} \rangle \in S$, the term $r^{-\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \star t_s = r^{-\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \star (r^s \star t)$ becomes equal to $(r^{-\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \cdot r^{\langle \boldsymbol{x}, \boldsymbol{k} \rangle}) \star t = t$. Therefore, $\mathsf{CEval}(\mathsf{ck}, \boldsymbol{x})$ is equal to the master secret key evaluation on all such inputs.

We now analyze the pseudorandomness of our CPRF. More precisely, any adversary that is given $\mathsf{ck}_{(S,\boldsymbol{k})}$ should not be able to efficiently distinguish the output of msk from random, when it is evaluated on an arbitrary challenge input $\boldsymbol{x}^*$ that satisfies $\langle \boldsymbol{x}^*, \boldsymbol{k} \rangle \notin S$. Similar to the Naor-Reingold CPRF of [19], the raw evaluation algorithm can be distinguished from random by any adversary that can ask even a single real evaluation output. To see why, first observe that due to the correctness of the construction, Eval can be re-written as

$$\mathsf{Eval}(\mathsf{msk}, \boldsymbol{x}) = \prod_{i=1}^{n} \alpha_i^{x_i} \cdot r^{\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \star t \ .$$

Any adversary can distinguish the output of this function from random by finding two different inputs $\boldsymbol{x} \neq \boldsymbol{x}'$ such that $\langle \boldsymbol{x}, \boldsymbol{k} \rangle = \langle \boldsymbol{x}', \boldsymbol{k} \rangle \notin S$. Given the output $y = \mathsf{Eval}(\mathsf{msk}, \boldsymbol{x})$, the adversary can compute $\mathsf{Eval}(\mathsf{msk}, \boldsymbol{x}')$ as $(\prod_{i=1}^{n} \alpha_i^{x'_i}) \cdot (\prod_{i=1}^{n} \alpha_i^{x_i})^{-1} \cdot y$, and therefore distinguish it from random.

Consequently, we focus on *no-evaluation* security. This guarantees that the CPRF behaves pseudorandomly on an unauthorized input $\boldsymbol{x}^*$ against an adversary who makes no evaluation queries. In fact, we start with an even weaker model, called *unpredictability*, in the no-evaluation setting. In this model, without making any evaluation queries, the adversary should not be able to *compute* the correct CPRF value for an unauthorized input. We later explain how to lift this to achieve full security.

Security Analysis. For our CPRF to be unpredictable, it should be hard, given the constrained key elements $(r^s \star t)_{s \in S}$ for an unknown $r \xleftarrow{\$} G$, to output $(v, r^v \star t)$ for some arbitrary $v \notin S$. While this problem does not directly appear in the literature, it can be related to a natural generalization of the Parallelization problem, which itself stands at the heart of group-action-based cryptography [29]. The Parallelization problem is the analogue of the computational Diffie-Hellman problem for group actions. When G is generated by a group element g_0, the worst-case Parallelization problem can be stated as follows: compute $g_0^{a+b} \star t$, given $g_0^a \star t$ and $g_0^b \star t$, where t is a public set element and a, b are unknown exponents. We investigate connections between this problem and our new problem, which we name power parallelization (Pow-Para) problem. To this end, we define an intermediate problem, the Parallelization problem with Auxiliary Inputs (Para-AI), that asks, given a set of group actions by the same *unknown* group element r to different *known* exponents $s_1, \ldots, s_m$, i.e. given $\{(s_1, r^{s_1} \star t), \ldots, (s_m, r^{s_m} \star t)\}$, to compute a new group action of the form

$r^{s_i+s_j} \star t$, such that $i \neq j$ are known. We show that this problem reduces to our problem for the exponent sets we are interested in.

We then compare Para-AI to a more standard hard problem called the Vectorization problem with Auxiliary Inputs (Vect-AI), introduced in [7] and analyzed in [39,38]. This problem generalizes the Vectorization problem to multiple inputs in a similar manner as Para-AI generalizes the Parallelization: the Vect-AI problem asks to find the *unknown* group element r, given its actions to different *known* exponents, i.e. given $\{(s_1, r^{s_1} \star t), \ldots, (s_m, r^{s_m} \star t)\}$, while in the Vectorization problem only the action $r \star t$ is provided. To emphasize the similarities between Para-AI and Vect-AI, we extend the quantum equivalence between Vectorization and Parallelization [40] to these problems, in the case where two actions are provided. Moreover, we believe that, in general, there are no other ways to attack Para-AI than to solve the corresponding Vect-AI problem. Hence, since the Vect-AI problem is a hard problem, according to [39,38], the Para-AI problem seems to be a reasonable hardness assumption for the security of our CPRF. We provide an overview of the analysis of this assumption in Section 4.4.

Instantiating from Isogenies. The CPRF presented in this work can be securely instantiated from any abelian effective group action (AEGA), see [3, Definition 3.4], for which the Pow-Para problem is hard. The notion of effective group action was introduced in [3], following [17,29], to formalize group actions where all the generic operations can be performed efficiently (e.g. sampling and multiplying group elements or evaluating actions). Currently, the most efficient frameworks to obtain AEGAs are provided by isogeny-based cryptography and the theory of oriented elliptic curves, due to [21] and generalized in [25]. In particular, the recent qt-`PEGASIS` [32,33] framework allows one to practically evaluate the action of *any* group element, which is crucial for our construction. Previous frameworks, such as [21,11,34], either restricted evaluation to specific group elements (they were only *restricted* efficient group actions) or were practical at low security levels only.[7] Thanks to the qt-`PEGASIS` framework, our CPRF can be efficiently instantiated to, what we believe to be, a security level of 128 bits, see Section 5.1. This security claim relies on our security analysis and the quantum analysis performed in [39].

To the best of our knowledge, the only other known constructions of group actions for cryptography, e.g., code-based group actions, do not offer *abelian* effective group actions.

Achieving Full Security. To lift our no-evaluation-secure CPRF to full security (where the adversary may query evaluations arbitrarily), we use a compiler inspired by [6]. The main idea is to apply a random oracle (RO) to the output of Eval (and CEval) to blind the correlations between the outputs. We improve this transformation in two aspects:

[7] Note that the first practical AEGA to overcome these limitations was introduced in [55].

1. First, we relax the required hardness of the base no-evaluation CPRF. We observe that it is sufficient for Eval to be *unpredictable* rather than pseudorandom. In other words, an adversary, given a constrained key (and no access to the evaluation oracle), should not be able to *compute* the exact value of Eval on an arbitrary unauthorized input. This observation in particular allows us to prove the security of our CPRF assuming the hardness of a *search* problem. The high-level idea stems from the fact that a random oracle has a one-way-to-random functionality, i.e., as long as it is hard for a PPT adversary to find x (unpredictability), the output $\mathsf{RO}(x)$ has full entropy to the adversary (pseudorandomness).
2. Secondly, the compiler of [6] requires the no-evaluation CPRF to be additionally *collision resistance*. We observe that this can be generically achieved by concatenating the input $\boldsymbol{x}$ to the Eval output (or the first 128 bits of it) before hashing, a technique also used in [58,2].

We additionally provide a security analysis of our CPRF in the quantum random oracle model (QROM). Although this analysis is slightly different from the ROM analysis, it leverages the same idea: as long as it is hard for a QPT adversary to issue a query with large amplitude toward x, the output $\mathsf{RO}(x)$ looks random to the adversary. In other words, if the adversary can distinguish $\mathsf{RO}(x)$ from random, then one can extract x by inspecting the queries of the adversary. This can be done by the One-Way to Hiding (O2H) lemma [4] which provides such an extractor. Our unpredictable-to-adaptive CPRF transformation together with the QROM analysis are provided in Section 4.2.

Efficiency Analysis. We estimate the efficiency of our final PCF construction, obtained by applying the BCMPR compiler to our isogeny-based CPRF together with the BIPSW weak PRF [13]. This weak PRF is defined as $F(\boldsymbol{k}, \boldsymbol{x}) = \lfloor \langle \boldsymbol{k}, \boldsymbol{x} \rangle \pmod 6 \rceil$, where $\boldsymbol{k}, \boldsymbol{x} \in \{0,1\}^n$, and the rounding function is defined as $\lfloor s \rceil = 0$ if $(s \bmod 6) \in \{0,1,2\}$, and $\lfloor s \rceil = 1$ if $(s \bmod 6) \in \{3,4,5\}$. The construction can be expressed as an IPM predicate defined by the set $S = \{k \mid 0 \leq k \leq n \text{ and } k = 0, 1, 2 \pmod 6\}$. Analysis of [23,46] suggest a key length of $n = 770$, which yields a set S of size $n/2 = 385$. In Section 5.2, we evaluate our construction based on the qt-`PEGASIS` instantiation of isogeny-based group actions for a security level of 128 bits. Although the receiver's key size is dominated by $|S|$ set elements, these elements can be viewed as public parameters that are independent of the wPRF key, and can be reused to establish PCF keys with multiple senders, which amortizes their memory cost over time. Moreover, we note that several memory optimizations used by [19] can also be applied to our construction to reduce the key sizes even more. More precisely, while our un-optimized PCF has a key size of around 146 kB for the sender and 243 kB for the receiver, we could reduce these to 16 Bytes for the sender and 103 kB for the receiver. We provide the details in Section 5.2.

Open Questions. Our work paves the way for multiple future possible directions to explore. We suggest some in the following:

– Can we build compact *public-key* PCFs (PK-PCFs) from isogenies? In a PK-PCF, the two involved parties participate in a non-interactive key exchange phase to derive their PCF keys, which is particularly useful in settings where any two pair of parties can participate in secure computation. Current known post-quantum PK-PCFs rely on lattices [26,2], which key sizes in the megabytes. To develop a public-key setup for our construction, one approach is to translate the group-based setup of [19] into group actions, which in particular requires developing a *distributed discrete logarithm* procedure, that is core of their setup, to group actions.

– Our CPRF can be viewed as a group-action extension of the Naor-Reingold CPRF of [19]. It remains intriguing to explore other directions of building a suitable CPRF for the BCMPR compiler from group actions.

– While assuming RO is standard in cryptography optimized for efficiency, theoretically, it remains interesting to remove the reliance on RO from our construction. As observed by previous works [6,19], a no-evaluation-secure CPRF can be made selectively-secure (a middle-level security for CPRFs) by applying a suitable *correlation-robust hash function* (CRH) to the output of Eval and CEval. We leave it as an open question to build a CRH from group actions or isogenies.

– The security of our CPRF is proved under the assumption that the Para-AI problem, introduced in this work, is hard. This problem reduces to the Vect-AI problem which has already been studied in the literature. The two problems are quantumly equivalent when only two inputs are provided, or arguably under some strong assumptions on the behavior of some oracle solving the Para-AI problem. We leave it as an interesting question to prove a general quantum equivalence between the Para-AI problem and the Vect-AI problem

3 Preliminaries

Notation. We use λ to denote the security parameter. For $n \in \mathbb{N}$, $[n]$ denotes $\{1, \ldots, n\}$. We mostly use bold lowercase letters (e.g., $\boldsymbol{r}$) to denote vectors. We write $\mathsf{poly}(\lambda)$, $\mathsf{superpoly}(\lambda)$, and $\mathsf{negl}(\lambda)$ to respectively denote an arbitrary polynomial, superpolynomial, and negligible function in λ. PPT stands for probabilistic polynomial time and QPT for quantum polynomial time. For a finite set S, $x \xleftarrow{\$} S$ denotes that x is sampled uniformly at random from S. For an algorithm $\mathcal{A}$, we denote by $y \leftarrow \mathcal{A}(x)$ the output y after running $\mathcal{A}$ on input x. For two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same domain, we write $\mathcal{D} \approx_c \mathcal{D}'$ to denote that they are computationally indistinguishable, and $\mathcal{D} \approx_s \mathcal{D}'$ to denote that they are statistically indistinguishable.

3.1 Pseudorandom Functions

We recall the definition of pseudorandom functions.

Definition 1 ((Weak) Pseudorandom Function (wPRF, PRF), [42,49]). *Let $\lambda \in \mathbb{N}$ be a security parameter. A (weak) pseudorandom function with domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, consists of the following two polynomial-time algorithms:*

- $\mathsf{KeyGen}(1^\lambda) \to k$*: A probabilistic algorithm that on input the security parameter λ, outputs a master secret key $k \in \mathcal{K}$.*
- $\mathsf{Eval}(k, x) \to y$*: A deterministic algorithm that on input the master secret key k, and an input value $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.*

We say that the pair $(\mathsf{KeyGen}, \mathsf{Eval})$ *is a*

- ***pseudorandom function (PRF)*** *if for all $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, it holds that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Eval}(k,\cdot)}(1^\lambda) = 1 \middle| k \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)\right] - \Pr\left[\mathcal{A}^{RF(\cdot)}(1^\lambda) = 1 \middle| RF \xleftarrow{\$} \mathcal{F}\right]\right| = \mathsf{negl}(\lambda),$$

where $\mathcal{F}$ is the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.

- ***weak pseudorandom function (wPRF)*** *if for all $\lambda \in \mathbb{N}$, any PPT adversary $\mathcal{A}$ and any polynomially bounded number $Q \in \mathbb{N}$, it holds that*

$$\left\{\left((x_i, \mathsf{Eval}(k, x_i))_{i\in[Q]}\right) \middle| \begin{array}{c} k \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda) \\ \forall i \in [Q] : x_i \xleftarrow{\$} \mathcal{X} \end{array}\right\} \approx_c \left\{\left((x_i, y_i)_{i\in[Q]}\right) \middle| \begin{array}{c} \forall i \in [Q] : \\ x_i \xleftarrow{\$} \mathcal{X},\ y_i \xleftarrow{\$} \mathcal{Y} \end{array}\right\}.$$

Definition 1 captures the PRF security notion that is referred to as the *real-or-random* security. We also recall the find-then-guess security of a pseudorandom function which is equivalent to the real-or-random security up to a multiplicative gap of $\mathcal{O}(Q)$ between the advantage functions, where Q is the number of evaluation queries.

Definition 2 (Find-then-Guess Security of (weak) PRFs). *Let λ be a security parameter. A function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is called a secure pseudorandom function if it is efficiently computable and the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible:*

- ***Setup.*** *The challenger chooses a random key $k \xleftarrow{\$} \mathcal{K}$ and a random bit $b \xleftarrow{\$} \{0,1\}$, and initializes a set $S = \varnothing$.*
- ***Pre-Challenge Evaluation Queries.*** *$\mathcal{A}$ adaptively sends arbitrary inputs $x \in \mathcal{X}$ to the challenger. The challenger computes and returns $F_k(x)$ to $\mathcal{A}$. It also updates $S \leftarrow S \cup \{x\}$.*
- ***Challenge Phase.*** *$\mathcal{A}$ sends an input $x^* \in \mathcal{X}$ as its challenge query to the challenger with the restriction that $x^* \notin S$. If $b = 0$, then the challenger computes $y^* \leftarrow F_k(x^*)$. If $b = 1$, then the challenger samples a random element $y^* \xleftarrow{\$} \mathcal{Y}$. It then returns y^* to $\mathcal{A}$.*
- ***Post-Challenge Evaluation Queries.*** *$\mathcal{A}$ continues sending arbitrary inputs $x \in \mathcal{X}$ to the challenger with the restriction that $x \neq x^*$, and receives $F_k(x)$.*
- ***Guess.*** *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

$\mathcal{A}$ wins if $b' = b$. We define the advantage of $\mathcal{A}$ in winning the game as $|2 \cdot \Pr[\mathcal{A}\ wins] - 1|$, where the probability is over the internal coins of $\mathcal{A}$ and the challenger.

3.2 Constrained Pseudorandom Functions

Definition 3 (Constrained Pseudorandom Functions). *Let λ be a security parameter. A constrained pseudorandom function (CPRF) with domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, that supports a class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, where each $C_\lambda \in \mathcal{C}_\lambda$ has domain $\mathcal{X}_\lambda$ and range $\{0,1\}$, consists of the following four polynomial-time algorithms:*[8]

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pp}, \mathsf{msk})$*: A probabilistic algorithm that on input the security parameter λ, outputs a public parameter pp and a master secret key $\mathsf{msk} \in \mathcal{K}$. We assume that all algorithms have access to pp.*
- $\mathsf{Eval}(\mathsf{msk}, x) \to y$*: A deterministic algorithm that on input the public parameter pp, the master secret key msk, and an input $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.*
- $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{ck}_C$*: A probabilistic algorithm that on input the master secret key msk, and a circuit $C \in \mathcal{C}$, outputs a constrained key ck_C.*
- $\mathsf{CEval}(\mathsf{ck}_C, x) \to y$*: A deterministic algorithm that on input the public parameter pp, a constrained key ck_C, and an input $x \in \mathcal{X}$, outputs a value $y \in \mathcal{Y}$.*

[8] In the rest of the paper, we drop the subscript λ when it is clear from context.

Correctness. *For all security parameter $\lambda \in \mathbb{N}$, any constrain $C \in \mathcal{C}$, and any input $x \in \mathcal{X}$ such that $C(x) = 0$, we have:*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) \neq \mathsf{CEval}(\mathsf{ck}_C, x) : \begin{array}{l}(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

1-Key Adaptive (Real-Or-Random) Security. *We say that a CPRF is 1-key adaptively secure if for all $\lambda \in \mathbb{N}$, the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible in λ:*

- ***Setup:*** *The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, initializes a set $S_{\mathsf{Eval}} = \varnothing$, and chooses a random bit $b \xleftarrow{\$} \{0,1\}$. It then (lazily) samples a random function $\mathcal{RF} : \mathcal{X} \to \mathcal{Y}$. It then sends pp to $\mathcal{A}$.*
- ***Pre-Constrained-Key Queries:*** *$\mathcal{A}$ can adaptively send arbitrary input values $x \in \mathcal{X}$ to the challenger. If $b = 0$, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$. Otherwise, if $b = 1$, the challenger computes $y \leftarrow \mathcal{RF}(x)$. It then returns y to $\mathcal{A}$, and updates $S_{\mathsf{Eval}} \leftarrow S_{\mathsf{Eval}} \cup \{x\}$.*
- ***Constrained-Key Query:*** *$\mathcal{A}$ can choose a circuit $C \in \mathcal{C}$ and sends it to the challenger. If there exists $x \in S_{\mathsf{Eval}}$ such that $C(x) = 0$, the challenger aborts and outputs $\perp$. Otherwise, the challenger computes $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ and returns the constrained key ck_C to $\mathcal{A}$.*
- ***Post-Constrained-Key Evaluation Queries:*** *$\mathcal{A}$ can continue submitting evaluation queries as before, with the restriction that it cannot query any x such that $C(x) = 0$.*[9]
- ***Guess:*** *$\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

We say that $\mathcal{A}$ wins if $b' = b$. We also define the advantage of $\mathcal{A}$ in winning a game as $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) := |2 \cdot \Pr[\mathcal{A}\ wins] - 1|$, where the probability is over the internal coins of $\mathcal{A}$ and the challenger.

1-Key Selective No-Evaluation Unpredictability. *We say that $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ is 1-key selective no-evaluation unpredictible if for all $\lambda \in \mathbb{N}$, the advantage of any PPT adversary $\mathcal{A}$ in the following game is negligible in λ:*

- ***Setup:*** *The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. It then sends pp to $\mathcal{A}$.*
- ***Constrained-Key Query:*** *The adversary chooses a circuit $C \in \mathcal{C}$ and sends it to the challenger. The challenger computes $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ and returns the constrained key ck_C to $\mathcal{A}$.*
- ***Guess:*** *$\mathcal{A}$ outputs a pair (x, y).*

We say that $\mathcal{A}$ wins if $C(x) = 1$ and $\mathsf{Eval}(\mathsf{msk}, x) = y$. We also define the advantage of $\mathcal{A}$ in winning this game as $\mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}}(\mathcal{A}) := \Pr[\mathcal{A}\ wins]$, where the probability is over the internal coins of $\mathcal{A}$ and the challenger.

[9] Equivalently, upon receiving an x such that $C(x) = 0$, the challenger, instead of aborting, can return $y \leftarrow \mathsf{CEval}(\mathsf{pp}, \mathsf{ck}_C, x)$

Remark 1. To the best of our knowledge, the unpredictibility property has not been used before and is introduced for the first time in our work. The more common notion in this context is 1-Key Selective no-evaluation pseudorandomness as follows. The adversary, after obtaining the constrained key, is given only one real or random sample and cannot distinguish between the two with non-negligible probability. We note that our search-type unpredictibility is a stronger security and therefore can be possbily based on weaker search-type assumptions.

In the following, we require that an (unbounded) adversary cannot predict the output of evaluation over an arbitrary chosen input, when it is only given the public parameters.

Definition 4 (Statistically Unpredictable CPRF). *A constrained pseudorandom function* $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ *is said to be* $\varepsilon_{\mathsf{up}}$*-statistically unpredictable if for any possibly unbounded algorithm* $\mathcal{A}$*, it holds that*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = y \mid (x, y) \leftarrow \mathcal{A}(\mathsf{pp})\right] \leq \varepsilon_{\mathsf{up}}(\lambda) \ ,$$

where $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*

3.3 Pseudorandom Correlation Functions

Definition 5 (Reverse-Sampleable Correlations). *Let* $\lambda \in \mathbb{N}$ *and* $1 \leq \ell_0(\lambda), \ell_1(\lambda) \leq \mathsf{poly}(\lambda)$ *be output-length functions. Let* $\mathcal{Y}$ *be a probabilistic algorithm that, on input* 1^λ*, returns a pair of outputs* $(y_0, y_1) \in \{0,1\}^{\ell_0(\lambda)} \times \{0,1\}^{\ell_1(\lambda)}$*, defining a correlation on the outputs. We say that* $\mathcal{Y}$ *defines a reverse-sampleable correlation if there exists a probabilistic polynomial time algorithm* $\mathsf{RSample}$ *which takes as input* 1^λ*,* $\sigma \in \{0,1\}$*, and* $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$*, and outputs* $y_{1-\lambda}^{\ell_{1-\sigma}(\lambda)}$*, such that for all* $\sigma \in \{0,1\}$ *the following distributions are statistically close:*

$$\{(y_0, y_1) \colon (y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)\}, \text{ and}$$
$$\{(y_0, y_1) \colon (y_0', y_1') \xleftarrow{\$} \mathcal{Y}(1^\lambda), y_\sigma \leftarrow y_\sigma', y_{1-\sigma} \leftarrow \mathsf{RSample}(1^\lambda, \sigma, y_\sigma)\} \ .$$

Definition 6 (OT Correlation). *A (1-out-of-2, n-bit) oblivious transfer (OT) correlation is a pair* $((r_0, r_1), (b, r_b))$*, where* $r_0, r_1 \xleftarrow{\$} \{0,1\}^n$*, and* $b \xleftarrow{\$} \{0,1\}$*. We denote by* $OT(1^\lambda, n)$ *the functionality that outputs a random (1-out-of-2, n-bit) OT-correlated pair.*

Remark 2 (OT Correlation is Reverse-Sampleable). *A (1-out-of-2, n-bit) OT correlation is reverse-sampleable.* $\mathsf{RSample}_{OT_n}(1^\lambda, \sigma, y_\sigma)$: If $\sigma = 0$, parse y_σ as $y_\sigma = (r_0, r_1)$, sample $b \xleftarrow{\$} \{0,1\}$, and output (b, r_b); Otherwise if $\sigma = 1$, parse y_σ as $y_\sigma = (b, r)$, sample $r' \xleftarrow{\$} \{0,1\}^n$, and output $((1-b)\cdot r + b \cdot r', b \cdot r + (1-b)\cdot r')$.

Next, we recall the definition of pseudorandom correlation functions. The following defines a weak PCF, where the properties are guaranteed on uniformly

sampled inputs. A strong PCF is defined similarly, where inputs can be sampled arbitrarily by the attacker. We note that any weak PCF can be transformed to a strong PCF in the random oracle model.

Definition 7 (Pseudorandom Correlation Functions (PCFs) [16]). *Let $\lambda \in \mathbb{N}$ be a security parameter, and $\mathcal{Y}$ be a reverse-sampleable correlation with output length functions $\ell_0(\lambda), \ell_1(\lambda)$. Let $\lambda \leq n(\lambda) \leq \mathsf{poly}(\lambda)$ be an input length function. A pseudorandom correlation function (PCF) consists of the following PT algorithms:*

- $\mathsf{PCF.KeyGen}(1^\lambda) \to (k_0, k_1)$*: A probabilistic algorithm that on input a security parameter λ outputs a pair of keys (k_0, k_1).*
- $\mathsf{PCF.Eval}(\sigma, k_\sigma, x) \to y_\sigma$*: A deterministic algorithm that takes as input a bit $\sigma \in \{0,1\}$, an evaluation key k_σ and an input value $x \in \{0,1\}^{n(\lambda)}$, and outputs $y_\sigma \in \{0,1\}^{\ell_\sigma(\lambda)}$.*

We say that a PCF for a correlation $\mathcal{Y}$ is (N, B, ϵ)-secure, if the following two conditions hold:

Pseudorandom $\mathcal{Y}$-Correlated Outputs. *For every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},1}(\lambda) = 1]\right| \leq \epsilon(\lambda),$$

where $\mathcal{A}$ makes at most $N(\lambda)$ queries to the oracle $\mathcal{O}^{\mathsf{pr}}_b$ in $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}$, that is defined in Experiment 1 for $b \in \{0,1\}$.

Security. *For every $\sigma \in \{0,1\}$ and every non-uniform adversary $\mathcal{A}$ of size $B(\lambda)$, it holds that for all sufficiently large λ,*

$$\left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,1}(\lambda) = 1]\right| \leq \epsilon(\lambda)$$

where $\mathcal{A}$ makes at most $N(\lambda)$ queries to the oracle $\mathcal{O}^{\mathsf{sec}}_b$ in $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}$, that is defined in Experiment 2 for $b \in \{0,1\}$.

Experiment 1 Pseudorandom $\mathcal{Y}$-Correlated Outputs

$\underline{\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},b}(\lambda)}$:

$\mathcal{Q} := \varnothing$
$(k_0, k_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{pr}}_b}(1^\lambda)$
Output b'

$\underline{\mathcal{O}^{\mathsf{pr}}_0}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
If $(x, y_0, y_1) \in \mathcal{Q}$**,**
  Return (y_0, y_1)
Else:
  $(y_0, y_1) \xleftarrow{\$} \mathcal{Y}(1^\lambda)$
  $\mathcal{Q} = \mathcal{Q} \cup \{(x, y_0, y_1)\}$
Return (x, y_0, y_1)

$\underline{\mathcal{O}^{\mathsf{pr}}_1}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
For $\sigma \in \{0,1\}$**:**
  $y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$
Return (x, y_0, y_1)

Experiment 2 PCF Security

RS is the algorithm for reverse sampling $\mathcal{Y}$.

$\underline{\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},\sigma,b}(\lambda)}$:

$(k_0, k_1) \leftarrow \mathsf{KeyGen}(\sigma)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{sec}}_b}(1^\lambda, \sigma, k_\sigma)$
Output b'

$\underline{\mathcal{O}^{\mathsf{sec}}_0}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
$y_{1-\sigma} \leftarrow \mathsf{Eval}(1-\sigma, k_{1-\sigma}, x)$
Return $(x, y_{1-\sigma})$

$\underline{\mathcal{O}^{\mathsf{sec}}_1}$:

$x \xleftarrow{\$} \{0,1\}^{n(\lambda)}$
$y_\sigma \leftarrow \mathsf{Eval}(\sigma, k_\sigma, x)$
$y_{1-\sigma} \leftarrow \mathsf{RS}(1^\lambda, \sigma, y_\sigma)$
Return $(x, y_{1-\sigma})$

3.4 Cryptographic Group Actions

Definition 8 (Group Actions). *A tuple $(G, T, \star)$ is a group action if G is a group, with neutral element denoted e_G, T is a set and $\star : G \times T \to T$ is a map such that*

- ***Identity.*** *For any $t \in T$, $e_G \star t = t$,*
- ***Compatibility.*** *For any $g_1, g_2 \in G$ and $t \in T$, $g_1 \star (g_2 \star t) = (g_1 g_2) \star t$.*

We say that G acts on T. Additionally, the action is regular if for any $t_1, t_2 \in T$, there exists a unique $g \in G$ such that $g \star t_1 = t_2$.

Group actions and their corresponding computational problems have been formalized for cryptographic purposes in several ways since their introduction in [17]. We follow the terminology of [3] to define a group action framework where all the basic operations are efficiently computable. Since, for the purpose of this work, we only consider abelian groups, we directly define group action frameworks assuming the commutativity of the acting groups.

Definition 9 (Abelian Effective Group Action (AEGA)). *Let $(G, T, \star)$ be a group action, where the group G and the set T are finite. We say that $(G, T, \star)$ is an abelian effective group action if G is an abelian group and there exist PPT algorithms solving the following problems:*

- ***Set (resp. Group) membership testing**. Decide if the input is a valid representation of a set (resp. group) element,*
- ***Equality testing**. Decide if two inputs represent the same group element,*
- ***Sampling**. Sample a random group element from a distribution indistinguishable from the uniform distribution over G,*
- ***Operating**. Compute the product of two given group elements,*
- ***Inverting**. Compute the inverse of a given group element,*
- ***Representing**. Compute a canonical bit string to represent a given set element,*
- ***Acting**. Compute the action of a group element on a set element.*

In addition, the set T is required to have a public element, denoted t_0.

To denote the hard problems at the core of our constructions, we consider the vectorization problem (VECTORIZATION) terminology from [29]. This problem can be seen as the analogue of the discrete-log problem in group actions.

Definition 10 (The Vectorization Problem (VECTORIZATION)). *Let $(G, T, \star)$ be a regular group action. The vectorization problem* (VECTORIZATION) *asks, given $t_1, t_2 \in T$, to compute the unique $g \in G$ such that $g \star t_1 = t_2$.*

Moreover, the analogue of the CDH problem in group actions is known as the parallelization problem (PARALLELIZATION) and is defined as below.

Definition 11 (The Parallelization Problem (PARALLELIZATION)). *Let $(G, T, \star)$ be a regular group action. The parallelization problem* (PARALLELIZATION) *asks, given $g_1 \star t$ and $g_2 \star t$, to compute the set element $g_1 g_2 \star t$.*

To guarantee their efficiency, the constructions presented in this work use AEGAs as frameworks. Then to obtain secure schemes, it is, at minima, required to consider AEGAs in which the PARALLELIZATION and the VECTORIZATION problems, and some of their variants, are hard. Nevertheless, the detailed security analysis performed in Section 4.4 is done assuming that the considered AEGAs are actually Cyclic Effective Group Actions with Twists (CEGAT), following the terminology of [1,38]. This choice is made to be as close as possible to the isogeny-based case where the acting group is virtually cyclic, see Section 4.3, and "twists" are computable.

The notion of twists comes from quadratic twists of elliptic curves. For a generic group action $(G, T, \star)$, we denote the twist as $\cdot^\top : T \to T : t \mapsto t^\top$ such that

$$\forall t \in T, \forall g \in G, (g \star t)^\top = g^{-1} \star t^\top.$$

Moreover, we assume that there exists an element t_0 in T, such that $(g \star t_0)^\top = g^{-1} \star t_0$, for any group element g.

Definition 12 (Cyclic Effective Group Action with Twists (CEGAT)). *Let $(G, T, \star)$ be an abelian effective group action. We say that $(G, T, \star, t_0)$ is a cyclic effective group action with twists if the group G is cyclic of publicly known order and public generator g_0 such that there exists a PPT algorithm solving the following problem:*

- ***Computing twists***. *Given $t := g \star t_0$, compute its twist $t^\top := g^{-1} \star t_0$.*

Remark 3. In this work, we define a CEGAT to differ from an AEGA not only by the cyclicity of the group and the possibility to compute twists, but also by the fact that the structure of the group is known, i.e. its order and a set of generators. Those properties are not required to instantiate the constructions we present here, assuming access to an AEGA is enough to efficiently run our schemes. Nevertheless, to have secure schemes, one needs to analyze their security assuming that adversaries have the maximal possible attack power. Hence, as argued in [38, §3.1], it can only improve the trustworthiness of the security analysis to assume that the attacker knows the structure of the group. In any case, once instantiated with oriented elliptic curves and isogenies, by [12], a quantum adversary can, under the Generalized Riemann Hypothesis, compute the structure of the group. The same reasoning applies to the computation of twists.

3.5 Elliptic Curves and Isogenies

We refer the reader to [60] for a comprehensive introduction to elliptic curves and their isogenies.

An *elliptic curve* E is an abelian variety of dimension 1. Namely, it is a smooth projective curve of genus 1 with a distinguished point ∞_E, called the point at infinity. For any prime $p > 3$ and any integer $n \geq 1$, an elliptic curve E defined over $\mathbb{F}_{p^n}$ can be given by a short Weierstrass equation $y^2 = x^3 + Ax + B$, for some $A, B \in \mathbb{F}_{p^n}$, such that $4A^3 + 27B^2 \neq 0$. In this work, we only consider elliptic curves defined over finite fields. For any extension $\mathbb{F}$ of $\mathbb{F}_{p^n}$, the set of $\mathbb{F}$-*rational points* is the set $\{(x, y) \in \mathbb{F}^2 \text{ such that } y^2 = x^3 + Ax + B\} \cup \{\infty_E\}$. The set of $\mathbb{F}$-rational points of E forms a commutative group with neutral element ∞_E.

An *isogeny* $\varphi : E \to E'$ of elliptic curves defined over $\mathbb{F}$ is a non-constant rational map sending ∞_E to ∞'_E. In particular, isogenies are maps between elliptic curves that preserve their group structure. The *degree* of an isogeny φ, denoted $\deg(\varphi)$, is its degree as a rational map. For any positive integer N, the *multiplication-by-N* map $[N] : E \to E$ is an isogeny of degree N^2. For any isogeny $\varphi : E \to E'$, there exists a unique isogeny $\hat{\varphi} : E' \to E$, called the *dual* isogeny, such that $\hat{\varphi} \circ \varphi = [\deg(\varphi)]$.

An isogeny from a curve to itself is an *endomorphism*. The set of all endomorphisms of an elliptic curve E together with the trivial map $[0] : P \mapsto \infty_E$ form a ring for point-wise addition and composition. This ring is called the *endomorphism ring* of E and is denoted $\mathrm{End}(E)$. An elliptic curve defined

over a finite field is *supersingular* if its endomorphism ring is not commutative, otherwise it is said to be *ordinary*. The *Frobenius endomorphism* of a curve E defined over $\mathbb{F}_{p^n}$ is the map $\pi_E : (x, y) \to (x^{p^n}, y^{p^n})$.

An isogeny of degree 1 is an *isomorphism*. Two elliptic curves E and E' defined over $\mathbb{F}_{p^n}$ are said to be isomorphic over an extension $\mathbb{F}$ of $\mathbb{F}_{p^n}$ if there exists an isomorphism between E and E' defined over $\mathbb{F}$. Two elliptic curves E and E' defined over $\mathbb{F}_{p^n}$ are isomorphic over the algebraic closure of $\mathbb{F}_{p^n}$, denoted $\bar{\mathbb{F}}_{p^n}$, if and only if $j(E) = j(E')$, where j denotes the *j-invariant*. For a curve E defined over $\mathbb{F}$ given in short Weierstrass form $E : y^2 = x^3 + Ax + B$, its j-invariant is $j(E) = 1728\frac{4A^3}{4A^3+27B^2} \in \mathbb{F}$. When E is a supersingular elliptic curve defined over a field of characteristic p, its j-invariant is defined over $\mathbb{F}_{p^2}$.

3.6 Oriented Elliptic Curves

For any quadratic order $\mathfrak{O}$, the class group of $\mathfrak{O}$, denoted $\mathrm{Cl}(\mathfrak{O})$, is the quotient group of the group of invertible fractional $\mathfrak{O}$-ideals by the subgroup of principal fractional $\mathfrak{O}$-ideals.

It is well-known that the class group $\mathrm{Cl}(\mathfrak{O})$ of an imaginary quadratic order $\mathfrak{O}$ acts regularly on the set of ordinary elliptic curves having their endomorphism ring isomorphic to $\mathfrak{O}$, see [61, Theorem 4.5]. This group action has been translated to sets of supersingular elliptic curves, notably in [35,21], and, then, generalized in [25] via the notion of orientations. The remainder of this subsection is also relying on [54].

Let $\mathfrak{O}$ be an imaginary quadratic order and E be a supersingular elliptic curve defined over a finite field $\mathbb{F}_{p^2}$. An $\mathfrak{O}$-*orientation* on the curve E is an injective morphism of rings $\iota : \mathfrak{O} \hookrightarrow \mathrm{End}(E)$. It is a *primitive* $\mathfrak{O}$-orientation when it can be extended to a superorder $\mathfrak{O}'$ of $\mathfrak{O}$ if and only if $\mathfrak{O}' = \mathfrak{O}$. We say that (E, ι) is a (primitively) $\mathfrak{O}$-oriented elliptic curve, when ι is a (primitive) $\mathfrak{O}$-orientation on E. Any isogeny $\varphi : E \to E'$ induces an orientation $(\varphi_*)(\iota)$ on its codomain as

$$(\varphi_*)(\iota)(\alpha) = \frac{\varphi \circ \iota(\alpha) \circ \hat{\varphi}}{\deg(\varphi)}, \forall \alpha \in \mathfrak{O}.$$

An $\mathfrak{O}$-oriented isogeny (resp. isomorphism) $\varphi : (E, \iota) \to (E', \iota')$ is an isogeny (resp. isomorphism) $\varphi : E \to E'$ such that $(\varphi_*)(\iota) = \iota'$. We define $\mathrm{SS}_{\mathfrak{O}}(p)$ to be the set of primitively $\mathfrak{O}$-oriented supersingular elliptic curve up to $\mathfrak{O}$-isomorphism. The set $\mathrm{SS}_{\mathfrak{O}}(p)$ is not empty if and only if p does not split in $\mathbb{Q} \otimes_{\mathbb{Z}} \mathfrak{O}$ and p does not divide the conductor of $\mathfrak{O}$.

3.7 Quantum Preliminaries

We refer the reader to [52] for an introduction to quantum states, measurements, and circuits.

For a function $f : \{0,1\}^n \to \{0,1\}^m$, a quantum oracle access to F is given by the unitary U_f that acts as $|x\rangle |y\rangle \to |x\rangle |y + f(x) \bmod 2\rangle$ for every $x \in \{0,1\}^n, y \in \{0,1\}^m$.

We require the following lemma in our proofs.

Lemma 1 (O2H Lemma from [4]). *Let $\mathcal{Z} \subseteq \mathcal{X}$ be a random set, h be a random bit-string, and $G, H : \mathcal{X} \to \mathcal{Y}$ be two random functions such that for every $x \notin \mathcal{Z}$ it holds that $H(x) = G(x)$ ($\mathcal{Z}, h, G, H$ may have arbitrary joint distribution).*

Let $\mathcal{A}$ be a quantum oracle algorithm with Q queries. Let $\mathcal{C}^H$ be an oracle algorithm that on input h does the following: it picks $i \xleftarrow{\$} [Q]$, runs $\mathcal{A}^H(h)$ until the i-th query, measures the query register in the computational basis, and ouputs the measurement outcome. Then it holds that

$$\left|\Pr[1 \leftarrow \mathcal{A}^H(h)] - \Pr[1 \leftarrow \mathcal{A}^G(h)]\right| \leq 2Q\sqrt{\Pr[z \in \mathcal{Z} \mid z \leftarrow \mathcal{C}^H(h)]}\,.$$

4 CPRFs from Group Actions

In this section, we construct an adaptively secure CPRF for inner-product memebership (IPM) predicates based on effective group actions, in the quantum random oracle model. We first define the IPM predicates. We then introduce an assumption on which we base our construction. Then propose our CPRF and show that that it is 1-key selective no-evaluation *unpredictable* under our assumption. Then, we introduce a transformation based on random oracles for boosting the security of the construction to obtain adaptive security. Finally, we analyze the security assumption when instantiated by the action of isogenies over elliptic curves.

Definition 13 (IPM Predicates). *Let $n \in \mathbb{N}$ and $S \subseteq [n]$. The class of inner-product membership (IPM) predicates parametrized by (n, S) is defined as $\mathsf{IPM}_{(n,S)} = \{C_{\boldsymbol{z}} \mid \boldsymbol{z} \in \{0,1\}^n\}$, where $C_{\boldsymbol{z}} : \{0,1\}^n \to \{0,1\}$ is such that $C_{\boldsymbol{z}}(\boldsymbol{x}) = 0$ iff $\langle \boldsymbol{x}, \boldsymbol{z} \rangle \in S$, for any $\boldsymbol{x} \in \{0,1\}^n$.*

4.1 A No-Evaluation Unpredictable CPRF for IPM Predicates

We define the following search problem and its corresponding hardness assumption.

Definition 14 ((n,S)-Power Parallelization Problem). *Let λ be a security parameter, $n \in \mathbb{N}$ and $S \subset [n]$. Moreover, let $(G, T, \star)$ be an AEGA per Definition 9. The (n,S)-power parallelization problem ((n,S)-Pow-Para) for $(G, T, \star)$ asks, given $(r^s \star t)_{s \in S}$, to find $(v, r^v \star t)$ for some $v \in [n] \setminus S$, where $t \xleftarrow{\$} T$ and $r \xleftarrow{\$} \mathbb{G}$.*

For any algorithm $\mathcal{A}$, we denote its success probability as $\mathsf{Adv}_{(n,S)\text{-Pow-Para}}(\mathcal{A})$, where the probability is over the internal coins of the problem and $\mathcal{A}$.

We say that the (n,S)-Pow-Para assumption holds for $(G, T, \star)$ if for any QPT $\mathcal{A}$ we have $\mathsf{Adv}_{(n,S)\text{-Pow-Para}}(\mathcal{A}) = \mathsf{negl}(\lambda)$.

For the moment, we assume that there exists an instantiation of the group action for which the above assumption holds and continue by introducing the

construction (the instantiation and the analysis of the assumption are deffered to Section 4.3 and 4.4). The construction is defined over binary inputs and constraint vectors, i.e., $\boldsymbol{x}, \boldsymbol{k} \in \{0,1\}^n$, and is presented in Construction 1.

Construction 1: No-Evaluation Unpredictable CPRF for IPM

- Let $n \in \mathsf{poly}(\lambda)$, $S \subset [n]$, and $(G, T, \star)$ be an AEGA.

$\mathsf{KeyGen}(1^\lambda)$:

- Sample $(g_0, \ldots, g_n) \xleftarrow{\$} G^{n+1}$.
- Let $t_0 \in T$, and define $t := g_0 \star t_0$. [a]
- Output $\mathsf{msk} := (t, g_1, \ldots, g_n)$.

$\mathsf{Constrain}(\mathsf{msk}, (S, \boldsymbol{k}))$:

- Sample $r \xleftarrow{\$} G$.
- $\forall s \in S$, define $t_s := r^s \star t$.
- $\forall i \in [n]$, compute $\alpha_i := r^{-k_i} \cdot g_i$.
- Output $\mathsf{ck}_{(S,\boldsymbol{k})} := ((t_s)_{s\in S}, \boldsymbol{\alpha}, \boldsymbol{k})$.

$\mathsf{Eval}(\mathsf{msk}, \boldsymbol{x})$:

- Output $(\prod_{i=1}^n g_i^{x_i}) \star t$.

$\mathsf{CEval}(\mathsf{ck}_{(S,\boldsymbol{k})}, \boldsymbol{x})$:

- Let $s_{\boldsymbol{x}} = \langle \boldsymbol{x}, \boldsymbol{k} \rangle$.
- If $s_{\boldsymbol{x}} \notin S$, output $\perp$ and abort.
- Output $(\prod_{i=1}^n \alpha_i^{x_i}) \star t_{s_{\boldsymbol{x}}}$.

[a] t_0 can be public

Theorem 1. *Let* CPRF *be as in construction 1. Then for any adversary* $\mathcal{A}$ *against the 1-key selective unpredictability of* CPRF*, there exists an algorithm* $\mathcal{B}$ *with the same runtime as* $\mathcal{A}$ *such that* $\mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}}(\mathcal{A}) \leq \mathsf{Adv}_{(n,S)\text{-Pow-Para}}(\mathcal{B})$*. Moreover,* CPRF *is perfectly correct.*

Proof. Correctness follows from inspection and by noting that G *is Abelian. We then prove unpredictability. Let* $\mathcal{A}$ *be a PPT adversary that can break unpredictability with non-negligible probability. We build a PPT* $\mathcal{B}$ *(with the same runtime as* $\mathcal{A}$*) that can attack the assumption presented in Definition 14. Before explaining* $\mathcal{B}$*, we first note that the constrained key* $\mathsf{ck}_{(S,\boldsymbol{k})}$ *can be sampled differently as follows. Sample* $r, g_0 \xleftarrow{\$} G$ *and* $t_0 \xleftarrow{\$} T$*, then set* $t := g_0 \star t_0$ *and* $t_s := r^s \star t$ *for all* $s \in S$*. Then sample* $\boldsymbol{\alpha} \xleftarrow{\$} G^n$ *and set* $g_i := \alpha_i \cdot r^{k_i}$*. Note that the view of the adversary remains identical with this modification since* $\boldsymbol{\alpha}$ *is uniformly random which itself comes from the fact that each* k_i *is a uniformly random group element. Then* $\mathcal{B}$ *proceeds as follows. It receives* $(\boldsymbol{k}, S)$ *from* $\mathcal{A}$*. It forwards* S *to the assumption oracle and receives* $(t_s)_{s\in S}$*. It then sets* $\mathsf{ck}_{(\boldsymbol{k},S)} = ((t_s)_{s\in S}, \boldsymbol{\alpha}, \boldsymbol{k})$ *as explained earlier and forwards the key to* $\mathcal{A}$*. Let* $(\boldsymbol{x}, y)$ *be the ouput of* $\mathcal{A}$*. By assumption, with non-negligible probablity it hold that* $\langle \boldsymbol{x}, \boldsymbol{k} \rangle \notin S$ *and* $\mathsf{Eval}(\mathsf{msk}, \boldsymbol{x}) = y$*. It follows that* $(g_1^{x_1} \cdot \cdots \cdot g_n^{x_n}) \star t = y$ *or equivalently* $(\alpha_1^{x_1} \cdot \cdots \cdot \alpha_n^{x_n}) \cdot r^{\langle \boldsymbol{x}, \boldsymbol{k} \rangle} \star t = y$*. Therefore* $\mathcal{B}$ *can break the assumption*

by outputting

$$\left(\langle \boldsymbol{x}, \boldsymbol{k} \rangle, (\alpha_1^{x_1} \cdot \cdots \cdot \alpha_n^{x_n})^{-1} \star y\right) ,$$

which concludes the proof.

□

On the choice of S. Recall that eventually we will use our CPRF for IPM to build PCF. This will be done by considering a weak PRF as the constraint of the CPRF, i.e., the choice of S depends on the following weak PRF. We use the weak PRF construction of [13]. This candidate is defined as $F(\boldsymbol{k}, \boldsymbol{x}) = \lfloor \langle \boldsymbol{k}, \boldsymbol{x} \rangle \pmod 6 \rceil$, where $\boldsymbol{k}, \boldsymbol{x} \in \{0,1\}^n$, and the rounding function is defined as $\lfloor s \rceil = 0$ if $(s \bmod 6) \in \{0,1,2\}$, and $\lfloor s \rceil = 1$ if $(s \bmod 6) \in \{3,4,5\}$. This weak PRF can also be expressed as an IPM predicate defined by the set $S = \{k \mid 0 \leq k \leq n \text{ and } k = 0,1,2 \pmod 6\}$, which would be our choice of S.

Definition 15. *For any $n \in \mathbb{N}$, we let* $\mathsf{BIPSW}(n) := \{k \mid 0 \leq k \leq n \text{ and } k = 0,1,2 \pmod 6\}$.

4.2 Achieving Adaptive Security

In this section, we propose a transformation that compiles a selective unpredictable CPRF into an adaptive pseudorandom CPRF. The transformation relies on a hash function. It modifies the evaluation by first concatenating the input and output, and then hashing the result, as detailed in Transform 1.

Transform 1: Unpredictability to Adaptive Pseudorandomness

- Let $\mathsf{CPRF}_{\mathsf{up}} = (\mathsf{KeyGen}_{\mathsf{up}}, \mathsf{Eval}_{\mathsf{up}}, \mathsf{Constrain}_{\mathsf{up}}, \mathsf{CEval}_{\mathsf{up}})$ be a CPRF, with input domain $\mathcal{X}$ and output range $\mathcal{Y}$, that supports a class of circuits $\mathcal{C}$ as constraints.
- Let $H : \mathcal{X} \times \mathcal{Y} \to \mathcal{T}$ be a hash function, where $\mathcal{T} \subseteq \{0,1\}^*$.

$\mathsf{KeyGen}(1^\lambda)$:
- $\mathsf{msk} \leftarrow \mathsf{KeyGen}_{\mathsf{up}}(1^\lambda)$.
- Output msk.

$\mathsf{Eval}(\mathsf{msk}, x)$:
- Output $H(x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x))$.

$\mathsf{Constrain}(\mathsf{msk}, C)$:
- $\mathsf{ck}_C \leftarrow \mathsf{Constrain}_{\mathsf{up}}(\mathsf{msk}, C)$.
- Output ck_C.

$\mathsf{CEval}(\mathsf{ck}_C, x)$:
- Output $H(x \| \mathsf{CEval}_{\mathsf{up}}(\mathsf{ck}_C, x))$.

We note that this is similar to the proposal of [6] with the exception that theirs does not perform the concatenation and simply hashes the evaluated output. We also adopt a different analysis approach which allows to rely on

a weaker assumption. More precisely, while they require CPRF to be 1-key selectively no-evaluation *pseudorandom*, we observe that *unpredictability* suffices. Moreover, we present a proof in the quantum random oracle model (QROM) for the resulting CPRF, while the analysis of [6] is limited to the random oracle model (ROM). We also note that Transformation 1 preserves the correctness.

Theorem 2. *Let $\varepsilon_{\mathsf{up}} \in [0,1]$ be a function of the security parameter λ. Let $\mathsf{CPRF}_{\mathsf{up}}$ be an $\varepsilon_{\mathsf{up}}$-statistically unpredictable CPRF (as per Definition 4). Let H be a hash function modeled as a random oracle. Let CPRF be the result of applying Transform 1 to $\mathsf{CPRF}_{\mathsf{up}}$ using H. Then for every adversary $\mathcal{A}$ attacking the 1-key adaptive security of CPRF, there exists an adversary $\mathcal{B}$ attacking the 1-key selective unpredictability of $\mathsf{CPRF}_{\mathsf{up}}$ such that*

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq Q_H \cdot Q_E \cdot \mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}_{\mathsf{up}}}(\mathcal{B}) + Q_H \cdot Q_E \cdot \varepsilon_{\mathsf{up}}\,,$$

where Q_H and Q_E are respectively the number of H queries and evaluation queries of $\mathcal{A}$. Moreover, $\mathcal{B}$ runs in time $\mathcal{O}(\mathcal{A})$.

Proof. *Let $\mathcal{A}$ be an adversary that breaks the adaptive security of CPRF by making Q_H random oracle queries and Q_E evaluation queries. Consider the following sequence of hybrid games:*

***Hybrid** $\mathcal{H}_0$. This is the real game of adaptive security, which is as follows:*

- ***Setup:** The challenger runs $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, initializes a set $S_{\mathsf{Eval}} = \varnothing$, and chooses a random bit $b \xleftarrow{\$} \{0,1\}$. It then (lazily) samples a random function $\mathcal{RF} : \mathcal{X} \to \mathcal{Y}$. It then sends pp to $\mathcal{A}$.*
- ***Pre-Constrained-Key Queries:** $\mathcal{A}$ can adaptively send arbitrary input values $x \in \mathcal{X}$ to the challenger. If $b = 0$, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$. Otherwise, if $b = 1$, the challenger computes $y \leftarrow \mathcal{RF}(x)$. It then returns y to $\mathcal{A}$, and updates $S_{\mathsf{Eval}} \leftarrow S_{\mathsf{Eval}} \cup \{x\}$.*
- ***Constrained-Key Query:** $\mathcal{A}$ can choose a circuit $C \in \mathcal{C}$ and sends it to the challenger. If there exists $x \in S_{\mathsf{Eval}}$ such that $C(x) = 0$, the challenger aborts and outputs $\perp$. Otherwise, the challenger computes $\mathsf{ck}_C \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$ and returns the constrained key ck_C to $\mathcal{A}$.*
- ***Post-Constrained-Key Evaluation Queries:** $\mathcal{A}$ can continue submitting evaluation queries as before, with the restriction that it cannot query any x such that $C(x) = 0$.*
- ***Guess:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$.*

***Hybrid** $\mathcal{H}_1$. This is similar to $\mathcal{H}_0$ except that the challenger aborts and outputs $\perp$ if the adversary $\mathcal{A}$, during the pre-constrained key queries, makes a query y to the random oracle and an evaluation query on input x such that $y = x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$. We let B denote this event.*

Hybrid $\mathcal{H}_1$ can be distinguished from $\mathcal{H}_0$ with probability at most $\Pr[B]$. We now compute this probability. Let $i^* \in [Q_H]$ and $j^* \in [Q_E]$ be respectively the indices of the random oracle query and the evaluation query on which B

occurs for the first time. We can correctly guess i^* and j^* with probability at least $1/(Q_H Q_E)$. Without loss of generality, assume that i^* query is issued after j^* query. Then we can break the statistical unpredictability of $\mathsf{CPRF}_{\mathsf{up}}$ as follows. An attacker $\mathcal{B}$ of the statistical unpredictability of $\mathsf{CPRF}_{\mathsf{up}}$ acts as the challenger of the adaptive security of CPRF. It answers the random oracle queries made by $\mathcal{A}$ using H, and answers the evaluation queries on any input x by returning $G(x)$ where G is another random oracle. It then halts at the i^*-th random oracle query of $\mathcal{A}$. Since the function $x \mapsto x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$ is injective, the view of $\mathcal{A}$ is identical to its real view up to this point. Consequently, $\mathcal{B}$ sets x to be the j^*-th evaluation query made by $\mathcal{A}$, and parses the i^*-th random oracle query as $x||y$. It then asks the unpredictability challenger for a constrained key for an arbitrary circuit C such that $C(x) = 1$, ignores the response, and outputs (x, y). Since it holds that $y = \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$, $\mathcal{B}$ succeeds in breaking the unpredictability of $\mathsf{CPRF}_{\mathsf{up}}$ with probability at least $\Pr[B] \cdot 1/(Q_H \cdot Q_E)$. This implies that $\Pr[B] \leq Q_H \cdot Q_E \cdot \varepsilon_{\mathsf{up}}$.

Hybrid $\mathcal{H}_2$***.*** *In this game, we reprogram the random oracle H for the evaluation queries. More precisely, when the adversary issues an evaluation query on input x, the challenger reprograms the random oracle H on $x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$ to output a uniformly random value.*

The adversary cannot distinguish this game from $\mathcal{H}_1$ except if it has made a random oracle query y and an evaluation query on x such that $y = x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$. Let E denote this event. By our conditioning from the previous game, E does not happen during the pre-constrained key phase. Therefore, such a query is issued after that $\mathcal{A}$ obtains the constrained-key and it must hold that $C(x) = 1$. The rest of the proof is similar to the argument of the previous game. More precisely, let $\mathcal{B}$ be an algorithm against the selective unpredictibility of $\mathsf{CPRF}_{\mathsf{up}}$. It runs $\mathcal{A}$ with H and G respectively for answering random oracle and evaluation queries. When $\mathcal{A}$ issues the constrained-key query, $\mathcal{B}$ passes the query to its own challenger, and relays the answer to $\mathcal{A}$. Up until E happens, this perfectly simulates $\mathcal{A}$'s view. Same as above, $\mathcal{B}$ can extract x and y such that $y = x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$ with $C(x) = 1$ with probability at least $1/(Q_H Q_E)$. Therefore, it holds that

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq Q_H \cdot Q_E \cdot \mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}_{\mathsf{up}}}(\mathcal{B}) \,.$$

Hybrid $\mathcal{H}_3$***.*** *In this game, the evaluation queries are responded by freshly sampled random elements from the range of H and not using Eval. The distribution remains perfectly identical to that of the previous game. Moreover, the view of the adversary is exactly the same in both cases of $b = 0$ and $b = 1$, and therefore the adversary has zero advantage in guessing b.*

□

In the proof of the above theorem, we extensively use random oracle reprogramming techniques. In the QROM, the reprogramming of the random oracle cannot be done as in the classical setting since the adversary makes quantum superposition queries to the random oracle. Therefore, we use the one-way to hiding Lemma (Lemma 1) for our argument. In simple words, this lemma

determines the probability that an adversary can detect a reprogramming of the random oracle. Using this lemma, we obtain the following result in the QROM.

Theorem 3. *Let $\varepsilon_{\mathsf{up}} \in [0,1]$ be a function of the security parameter λ and $\mathsf{CPRF}_{\mathsf{up}}$ be a CPRF that is $\varepsilon_{\mathsf{up}}$-statistically unpredictable (per Definition 4). Let H be a hash function modeled as quantum random oracle and CPRF be defined as above. Then for every quantum adversary $\mathcal{A}$ in the 1-key adaptive security game of CPRF, there exists a quantum adversary $\mathcal{B}$ against the 1-key selective unpredictability of $\mathsf{CPRF}_{\mathsf{up}}$ such that*

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq 2Q_H \sqrt{(\mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}_{\mathsf{up}}}(\mathcal{B}) + \varepsilon_{\mathsf{up}})}\,,$$

where Q_H is the number of H queries of $\mathcal{A}$. Moreover, $\mathcal{B}$ runs in time $\mathcal{O}(\mathcal{A})$.

Proof. Consider the following sequence of hybrid games:

Hybrid $\mathcal{H}_0$**.** *It is the real game of adaptive security.*

Hybrid $\mathcal{H}_1$**.** *It is similar to $\mathcal{H}_0$, except that we reprogram H on every input of the form $x \| \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)$ for all $x \in \mathcal{X}$.*

We bound the probability that $\mathcal{A}$ can distinguish $\mathcal{H}_1$ from $\mathcal{H}_0$ by using the O2H lemma 1. Let $\mathcal{Z}$ be the following set: $\{x \| y \in \mathcal{X} \times \mathcal{Y} \mid y = \mathsf{Eval}_{\mathsf{up}}(\mathsf{msk}, x)\}$. Let G be a function such that $G(z) := H(z)$ for all $z \notin \mathcal{Z}$ and $G(z) \xleftarrow{\$} \mathcal{T}$ for all $z \in \mathcal{Z}$. By O2H Lemma 1, for every algorithm $\mathcal{A}$ there exists an algorithm $\mathcal{C}$ such that

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}'}(\mathcal{A}) \leq 2Q_H \sqrt{\Pr(z \in \mathcal{Z} \mid z \leftarrow \mathcal{C}^H)}\,,$$

where $\mathcal{C}$ runs in time $\mathcal{O}(\mathcal{A})$. Now note that $\mathcal{C}^H$ measures $\mathcal{A}$ at the i-th oracle query for a randomly chosen $i \xleftarrow{\$} [Q_H]$. Depending on whether the i-th query issued before the constrained-key query of $\mathcal{A}$ or after, one can respectively bound $\Pr(z \in \mathcal{Z} \mid z \leftarrow \mathcal{C}^H)$ by statistical unpredictability or computational unpredictability. Therefore, it holds that

$$\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq 2Q_H \sqrt{(\mathsf{Adv}^{\mathsf{up}}_{\mathsf{CPRF}_{\mathsf{up}}}(\mathcal{B}) + \varepsilon_{\mathsf{up}})}\,,$$

for some $\mathcal{B}$ with the same runtime as $\mathcal{O}(\mathcal{C})$.

Hybrid $\mathcal{H}_3$**.** *In this game, we answer the evaluation queries using freshly sampled elements and ignore Eval. The distribution remains perfectly identical to that of the previous game. The adversary has zero advantage to guess the bit b.*

□

Remark 4 (Discussion on ROM vs QROM). Although the O2H lemma considers QROM, but ROM variants are also available. More precisely, the exact statement holds in the ROM with the tighter upperbound $Q_H \cdot \Pr[z \in \mathcal{Z} \mid z \leftarrow \mathcal{C}^H(h)]$. The proof is immediate; any adversary that can detect the reprogramming must have queried the reprogrammed input. This yields even a tighter reduction for the ROM compared to Theorem 2 (Q_E times smaller). However since for practical applications Q_E is orders of magnitude smaller than Q_H, this improvement is not significant. Therefore, we keep both statements for completeness.

Finally, by applying the transform to our initial no-evaluation unpredictable construction, one obtains the following corollary.

Corollary 1. *Let* $\mathsf{CPRF}_{\mathsf{up}}$ *be as in Construction 1 instantiated with parameters* n, S*, and a regular AEGA* $(G, T, \star)$*. Let* H *be a hash function modeled as a (quantum) random oracle. Let* CPRF *be obtained by applying Transform 1 to* $\mathsf{CPRF}_{\mathsf{up}}$ *and* H*. Then for every adversary* $\mathcal{A}$ *against the 1-key adaptive pseudorandomness of* CPRF*, there exists an adversary* $\mathcal{B}$ *against the 1-key no-evaluation unpredictability of* $\mathsf{CPRF}_{\mathsf{up}}$ *with the same runtime as* $\mathcal{A}$*, such that*

- *In the ROM, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq Q_H \left(\mathsf{Adv}_{(n,S)\text{-Pow-Para}}(\mathcal{B}) + 1/\#G\right)$.
- *In the QROM, we have* $\mathsf{Adv}^{\mathsf{adp}}_{\mathsf{CPRF}}(\mathcal{A}) \leq 2Q_H \sqrt{\mathsf{Adv}_{(n,S)\text{-Pow-Para}}(\mathcal{B}) + 1/\#G}$.

Proof. The proof is immediate according to Theorem 1, 2, and 3. One only has to note that $\varepsilon_{\mathsf{up}} = 1/\#G$ *for any regular AEGA.* □

Corollary 2. *Let* λ *be a security parameter. Let* $\mathsf{CPRF}_{\mathsf{up}}$ *be as in Construction 1 instantiated with parameters* n, S*, and a regular AEGA* $(G, T, \star)$*. Let* H *be a hash function modeled as a (quantum) random oracle. Let* CPRF *be obtained by applying Transform 1 to* $\mathsf{CPRF}_{\mathsf{up}}$ *using* H*. Then* CPRF *is an adaptively-secure CPRF (as per Definition 3) for IPM predicates defined by parameters* (n, S) *if* (n, S)-Pow-Para *problem (Definition 14) is hard, and* $\#G = \mathsf{superpoly}(\lambda)$*.*

4.3 Instantiation of the Group Action

In this section, we propose a group action to instantiate Construction 1 and (n, S)-Pow-Para. For the moment, we limit the discussion to the choice of the group action and the structure that it offers (whether it is AEGA or CEGAT or something else). The concrete implementation, the choice of parameters and runtime/memory complexity are reserved for the final PCF for OT construction and discussed in Section 5.

What group action do we choose? Let $\mathfrak{O}$ be an imaginary quadratic order, $\mathrm{Cl}(\mathfrak{O})$ be its ideal class group, and $\mathrm{SS}_{\mathfrak{O}}(p)$ be the set of primitively $\mathfrak{O}$-oriented supersingular elliptic curve over $\mathbb{F}_{p^2}$ up to $\mathfrak{O}$-isomorphism. From any $\mathfrak{O}$-oriented elliptic curve (E, ι) and any integral $\mathfrak{O}$-ideal $\mathfrak{a}$, one can build an $\mathfrak{O}$-oriented isogeny $\varphi_{\mathfrak{a}} : (E, \iota) \to (E^{\mathfrak{a}}, (\varphi_{\mathfrak{a}})_*(\iota))$ of degree equal to the norm of $\mathfrak{a}$ (see [54, Proposition 3.5] for more details). When (E, ι) is primitively $\mathfrak{O}$-oriented and $\mathfrak{a}$ is invertible, (E', ι') is also primitively $\mathfrak{O}$-oriented. This construction provides an action from the class group $\mathrm{Cl}(\mathfrak{O})$ on the set of primitively $\mathfrak{O}$-oriented curve $\mathrm{SS}_{\mathfrak{O}}(p)$ with at most two orbits. We denote this action as

$$\begin{aligned} \star : \mathrm{Cl}(\mathfrak{O}) \times \mathrm{SS}_{\mathfrak{O}}(p) &\to \mathrm{SS}_{\mathfrak{O}}(p) \\ ([\mathfrak{a}], (E, \iota)) &\mapsto [\mathfrak{a}] \star (E, \iota) := (E^{\mathfrak{a}}, (\varphi_{\mathfrak{a}})_*(\iota)) \end{aligned}$$

Once restricted to one of its orbit, the group action $(\mathrm{Cl}(\mathfrak{O}), \mathrm{SS}_{\mathfrak{O}}(p), \star)$ is regular.

We use the regular variant of the above group action to instantiate the construction and the assumption. The efficiency of this instantiation depends

on the concrete algorithms for computing the operations of the group (e.g. multiplication or inverse) and the action of the group on the set, as well as the choice of the parameters (the imaginary quadratic order $\mathfrak{O}$, the characteristic p of the field and the degree of the extension n). Note that, once implemented at low-level, the runtime of group operations should be negligible compared to the cost of group action evaluations. Many efforts have been made to obtain practical algorithms for performing these actions: from the `CSIDH` framework [21], which only allows for the evaluation of actions by specific small norm ideals[10], to the recent qt-`PEGASIS` framework that we use in this work. The latter provides an AEGA structure for our chosen group action and currently achieves the best runtime performance.

Since it does not impact the rest of this work, we do not provide details on the algorithm used in qt-`PEGASIS` to evaluate group actions. Henceforth, we treat it as black-box. Nevertheless, it is important to describe more explicitly the group and the set considered in the qt-`PEGASIS` framework as it underlies crucial aspects of our constructions, such as their security and complexity.

First of all, when we are referring to this algorithm, we are actually referring to its instantiation following the "`CSURF` context" [20], as used in [32,33,31] to obtain practical implementations of `PEGASIS`. This means that we consider the group action $(\mathrm{Cl}(\mathfrak{O}), \mathrm{SS}_{\mathfrak{O}}(p), \star)$ where

- $p := c2^f - 1$ for a large f and a small c, such that $p \equiv 7 \bmod 8$,
- $\mathfrak{O} := \mathbb{Z}[\frac{1+\sqrt{-p}}{2}]$.

In this case, just as in `CSIDH` or `CSURF`, every primitively oriented supersingular elliptic curve is actually defined over $\mathbb{F}_p$ and the orientation is always given by the Frobenius endomorphism. Hence, the elements of $\mathrm{SS}_{\mathfrak{O}}(p)$ are entirely characterized by their j-invariant, which are defined over $\mathbb{F}_p$. This leads to a very compact representation for the set elements of $\log p$ bits only.

Another important observation on this framework is that the discriminant of the order $\mathfrak{O}$ is $-p$. Then, by Brauer–Siegel theorem, the order of the class group $\mathrm{Cl}(\mathfrak{O})$ is close to $\sqrt{p}$, for large enough p. This value is crucial to estimate the hardness of the Vectorization and Parallelization problems, as well as their variants, for the group action $(\mathrm{Cl}(\mathfrak{O}), \mathrm{SS}_{\mathfrak{O}}(p), \star)$. We detail this analysis in Section 5.1.

On the other hand, it yields an estimation on the bit size of the group elements. By [37], one can store a representative of an ideal class of an imaginary quadratic order O in $\frac{3}{2}\log \mathrm{ord}(\mathrm{Cl}(O))$ bits. Hence, we can here store group elements of $\mathrm{Cl}(\mathfrak{O})$ using around $\frac{3}{4}\log p$ bits.

More information on the chosen group action is also known:

1. First, the structure of the group $\mathrm{Cl}(\mathfrak{O})$ is computable in quantum polynomial time, under the Generalized Riemann Hypothesis [12].
2. Second, the twists of actions on E_0, with $j(E_0) = 1728$, can be computed efficiently from quadratic twists (see [21]).

[10] Therefore, `CSIDH` does not satisfy the properties of Definition 9 and is not an AEGA (it is only a *Restricted* AEGA, see [3, Definition 3.8]).

3. Third, the heuristic of Cohen-Lenstra [24] asserts that imaginary class groups are cyclic $\approx 97\%$ of the times. Moreover, it implies that when they are not cyclic, they have, with high probability, a cyclic part of nearly the same order than the whole group.

Consequently, under reasonable assumptions[11], the tuple $(\mathrm{Cl}(\mathfrak{O}), \mathrm{SS}_{\mathfrak{O}}(p), \star, E_0)$, defined as above, is a regular CEGAT for a quantum party.

4.4 Assumption Analysis

We analyze the hardness of (n, S)-Pow-Para with respect to the specific instantiation of the group action that was discussed in Section 4.3. To this end, we establish reductions between this problem and more familiar ones. As mentioned earlier, the considered instantiation provides a regular CEGAT structure. Therefore, in all of this section, we restrict the analysis to this framework. Moreover, we use the specific choice of $S = \mathsf{BIPSW}(n)$, see Definition 15.

We recall the following variant of the Vectorization problem.

Definition 16 (The S-Vectorization Problem with Auxiliary Inputs). *Let $(G, T, \star, t_0)$ be a CEGAT. The S-vectorization problem with auxiliary inputs (S-Vect-AI) asks: Given $(s, r^s \star t_0)_{s \in S}$, where r is an unknown group element of G, find r.*

The S-Vect-AI problem, in the context of isogeny-based cryptography, was first introduced in [7] and was further studied by [38]. One can consider it as the group action analogue of the discrete-logarithm problem with auxiliary inputs [47]. Recent work in [39] provides an extensive analysis of this assumption and its hardness.

We also introduce the following problem.

Definition 17 (The S-Parallelization Problem with Auxiliary Inputs). *Let $(G, T, \star, t)$ be a CEGAT. The S-parallelization problem with auxiliary inputs (S-Para-AI) asks: Given $(s, r^s \star t_0)_{s \in S}$, where r is an unknown group element of G, compute $(v, r^v \star t_0)$ such that $v \notin S$ and $v = s + s'$ for some $s \neq s'$ in S.*

To the best of our knowledge, the S-Para-AI problem has not been introduced in the literature before, even though it arises as a natural variant of the S-Vect-AI problem or generalization of the Parallelization problem.

In the remainder of this section, we study the hardness of this problem. Then we prove the following theorem, which establishes a lower bound on the hardness of the (n, S)-Pow-Para used in Section 4.

[11] Assuming the Generalized Riemann Hypothesis is standard in the field. The Cohen-Lenstra heuristic is also common in the context of group actions in isogeny-based cryptography.

Theorem 4. *Let $n \equiv 2 \bmod 6$ be an integer and $S := \mathsf{BIPSW}(n)$. Then $(\{-3\} \cup S)$-Para-AI reduces to (n, S)-Pow-Para in polynomial time.*

We now turn to the relation between the S-Vect-AI problem and the S-Para-AI problem. Note that for the case where S is a singleton $\{a\}$, for a an integer coprime to the order of the group, we recover the original Vectorization problem from the S-Vect-AI problem. Recall that the Vectorization problem is quantumly equivalent to the Parallelization problem [40]. Therefore, we focus on the case where $\#S = 2$ as the simplest non-trivial extension. We show that Vectorization reduces to S-Vect-AI as follows:

Proposition 1. *The* Vectorization *problem reduces to the S-Vect-AI problem in polynomial time, where $S = \{-a, a\}$ for any arbitrary integer a that is coprime to $\mathrm{ord}(G)$.*

Proof. Let $(g \star t_0)$ be an instance of the Vectorization *problem. Let us denote by g_0 the publicly known generator of G. Then there exists a group element s such that $g = g_0^s$. By the properties of CEGATs, one can efficiently compute $g_0^{-s} \star t_0$ from $g_0^s \star t_0$. Since $\mathrm{ord}(G)$ is known, one can compute an integer a coprime to $\mathrm{ord}(G)$. Note that the pair $((a, g_0^s \star t_0), (-a, g_0^{-s} \star t_0))$ is a well-defined instance of the S-Vect-AI problem for $S = \{a, -a\}$. Indeed, since a is coprime to $\mathrm{ord}(G)$, the group element $h := g_0^{s/a}$ is well-defined and we have $g_0^s \star t_0 = h^a \star t_0$ and $g_0^{-s} \star t_0 = h^{-a} \star t_0$. Then, solving the S-Vect-AI problem for the input $((a, g_0^s \star t_0), (-a, g_0^{-s} \star t_0))$ provides the element h. Finally, by computing h^a, we get g as wished.* □

Moreover, there exists a special type of reduction from Vect-AI to Para-AI as shown below.

Proposition 2. *Given oracle access to S'-Para-AI for every set S' with two elements, one can solve the S-Vect-AI problem in quantum polynomial time, for any given set S with two elements $\{s, s'\}$ such that at least one of them is coprime to $\mathrm{ord}(G)$.*

Proof. Let $((s, r^s \star t_0), (s', r^{s'} \star t_0))$ be an instance of S-Vect-AI with the properties stated above. We show how to find r given the oracle access to S'-Para-AI for every set S' with two elements. Our proof follows the same argument as in [40, Theorem 1]. Recall that G, as the group acting in a CEGAT, is cyclic such that a generator g_0 is known. Therefore, there exists $z \in \mathbb{Z}$ such that $r = g_0^z$ and our goal is to find z. Since s or s' is coprime to $\mathrm{ord}(G)$, without loss of generality, one can assume that $\gcd(s, \mathrm{ord}(G)) = 1$. Consider the following function:

$$\begin{aligned} f : \mathbb{Z}^2 &\to T \\ (x, y) &\mapsto g_0^x \star (r^{sy} \star t_0) \end{aligned}$$

Note that f defines an abelian hidden subgroup problem instance with respect to the lattice

$$L = \{(x, y) \in \mathbb{Z}^2 \mid g_0^{x+zsy} \star t_0 = t_0\}\,.$$

Assuming that one can efficiently evaluate f, Shor's algorithm [59] finds the hidden group in quantum polynomial time. Then any element $(x, 1)$ of this group yields $g_0^{-x/s} \star t_0 = g_0^z \star t_0$, from which one can find z since s is known and $\gcd(s, \mathrm{ord}(G)) = 1$. So our task reduces to show how to evaluate f, given the oracle access to S'-Para-AI.

The action of g_0^x can be efficiently computed since both g_0 and x are known. For the action of $(r^s)^y$, as shown in [40, Lemma 1], it suffices to perform the double-and-add algorithm in the "implicit" group of the exponent. By adding, we mean obtaining $(a + b, r^{a+b} \star t_0)$ given $(a, r^a \star t_0)$ and $(b, r^b \star t_0)$. By doubling, we mean obtaining $(2a, r^{2a} \star t_0)$ given $(a, r^a \star t_0)$.

Addition: This can be done by calling the oracle $\{a, b\}$-Para-AI with the input $\{(a, r^a \star t_0), (b, r^b \star t_0)\}$.

Doubling: Note that, one cannot query $\{a, a\}$-Para-AI to obtain $(2a, r^{2a} \star t_0)$, since $\{a, a\}$ is not a set with 2 elements and, in any case, the S-Para-AI problem does not accept answers of the form $(s + s', r^{s+s'})$ for $s = s'$. Therefore, we proceed differently. We query $\{a, b\}$-Para-AI with input $\{(a, r^a \star t), (b, r^b \star t_0)\}$ for $b \neq a$ (such pair is known by assumption) to obtain the pair $(a+b, r^{a+b} \star t_0)$. Then, we query $\{a, a+b\}$-Para-AI with the input $((a, r^a \star t_0), (a+b, r^{a+s} \star t_0))$ to obtain $(2a+b, r^{2a+b} \star t_0)$. Finally, since $(G, T, \star, t_0)$ is a CEGAT, from the pair $(b, r^b \star t_0)$, one can compute the pair $(-b, r^{-b} \star t_0)$. Hence, with a last query to $\{2a + b, -b\}$-Para-AI with the input $((2a+b, r^{2a+b} \star t_0), (-b, r^{-b} \star t_0))$, we obtain the element $(2a, r^{2a} \star t_0)$.

This completes the proof. □

We summarize all these findings in Figure 1.

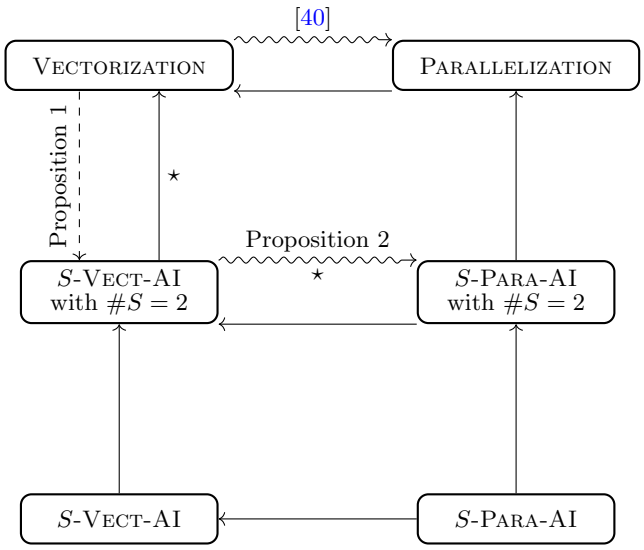

Fig. 1. Summary of reductions between group action cryptographic computational problems when instantiated with a CEGAT. Each arrow is a polynomial time reduction. Straight arrows are classical reductions. Snake arrows are quantum reductions which assume an access to an oracle solving any instance of the problem we are reducing to, e.g. the reduction from S-Vect-AI to S-Para-AI, with $\#S = 2$, assumes that one can solve any given instance of the S'-Para-AI problem for any given set S' with two elements. Dashed arrows are reductions that require only an access to an oracle solving the involved problem for a specific families of sets S. Arrows labeled with $\star$ assume that at least one element of S is coprime to the order of the considered acting group. Plain arrows are trivial reductions.

Proof of Theorem 4. We require the following lemma which states that the (n, S)-Pow-Para problem is random self-reducible.

Lemma 2. *Let n be an integer, S be a subset of $[n]$, $(G, T, \star, t_0)$ be a CEGAT, r_1 be a group element in G and t_1 be a set element in T. Then, there is a PPT algorithm that, given $(s, r_1^s \star t_1)_{s\in S}$, returns $(s, r^s \star t)_{s\in S}$, where $r \xleftarrow{\$} G$ and $t \xleftarrow{\$} T$, together with two group elements $(h_1, h_2) \in G^2$ such that $h_1 r = r_1$ and $h_2 \star t = t_1$.*

Proof. Let $n, S, (G, T, \star, t_0), r_1$ and t_1 be defined as in the above lemma. Using the CEGAT properties, one can sample a group element $g_1 \in G$ uniformly at random efficiently. We denote by r the group element $g_1 r_1$. Since r_1 is fixed and g_1 is sampled uniformly at random, the distribution that follows the group element r is also uniform. Moreover, for any $s \in S$, we have

$$g_1^s \star (r_1^s \star t_1) = (g_1 r_1)^s \star t_1 = r^s \star t_1.$$

We now sample a second group element $g_2 \in G$ uniformly at random. Then the set element $t := g_2 \star t_1$ follows a uniform distribution as well. In addition, for any $s \in S$, we have that $g_2 \star (r^s \star t_1) = r^s \star (g_2 \star t_1) = r^s \star t$. Hence, one can efficiently compute

$$(s, g_2 g_1^s \star (r_1^s \star t_1))_{s \in S} = (s, r^s \star t)_{s \in S},$$

where $r \xleftarrow{\$} G$ and $t \xleftarrow{\$} T$. By setting $h_1 := g_1^{-1}$ and $h_2 := g_2^{-1}$, we get two group elements such that $h_1 r = r_1$ and $h_2 \star t = t_1$. □

Let $(s, r_0^s \star t_0)_{s \in (\{-3\} \cup S)}$ be an instance of the $(\{-3\} \cup S)$-Para-AI problem. By Lemma 2, one can randomize this set of group actions to obtain a well-defined instance of the (n, S)-Pow-Para Problem. We let $(s, r^s \star t)_{s \in S}$ denote this instance and let h_1, h_2 denote the group elements such that $h_1 r = r_0$ and $h_2 \star t = t_0$. By solving it, we get $(v, r^v \star t)$ such that $v \in [n] \setminus S$. If there exists a pair $(s, s') \in (\{-3\} \cup S)^2$ such that $v = s + s'$ and $s \neq s'$, then

$$(v, (h_2 h_1^v) \star (r^v \star t)) = (v, r_0^v \star t_0)$$

would be the solution to the $(\{-3\} \cup S)$-Para-AI problem. By definition of $S := \mathsf{BIPSW}(n) = \{0 \leq k \leq n \text{ such that } k \equiv 0, 1, 2 \bmod 6\}$, we have that $v \in \{0 \leq k \leq n \text{ such that } k \equiv 3, 4, 5 \bmod 6\}$. For any congruence of v, there are indeed suitable choices for the integers s, s'. For example:

- If $v \equiv 3 \bmod 6$, we take $(s, s') := (v - 1, 1)$,
- If $v \equiv 4 \bmod 6$, we take $(s, s') := (v + 3, -3)$,
- If $v \equiv 5 \bmod 6$, we take $(s, s') := (v + 3, -3)$.

We now give more details on why $v + 3 \in S$ when $v \equiv 5 \bmod 6$. Since $v + 3 \equiv 2 \bmod 6$, we have that $v + 3$ is in $S = \mathsf{BIPSW}(n)$ provided $v + 3 \leq n$. From $v \equiv 5 \bmod 6$ and $n \equiv 2 \bmod 6$, we get

$$n - v \equiv 2 - 5 \equiv 3 \bmod 6.$$

As $n \geq v$, the difference $n - v$ is a nonnegative integer congruent to 3 mod 6, so we have $v + 3 \leq n$. Therefore $v + 3$ is in S. Note that the reasoning holds for $v \equiv 4 \bmod 6$ as well. By short congruence computations, we complete the verification that the pairs (s, s') above lie in $(\{-3\} \cup S)^2$. This completes the proof of Theorem 4

Conclusion. From Theorem 4, the $(\{-3\} \cup \mathsf{BIPSW}(n))$-Para-AI problem provides a lower bound for the hardness of $(n, \mathsf{BIPSW}(n))$-Pow-Para. Note that the latter is the assumption corresponding to our construction instantiation. Hence, to bound its hardness, it suffices to understand the hardness of $(\{-3\} \cup \mathsf{BIPSW}(n))$-Para-AI.

We provide two observations:

– First, Proposition 1 and 2 show that Vectorization can be solved given oracle access to S'-Para-AI for every set S' with two elements. This shows

that if Vectorization is hard then there are sets S' with $\#S' = 2$ such that S'-Para-AI is hard. Therefore, at least some variants of Para-AI are as hard as Vectorization. However, $(\{-3\} \cup \mathsf{BIPSW}(n))$-Para-AI is not necessarily one of the hard variants.
- Second, Para-AI is a natural variant of Vect-AI the same way that the CDH problem is a natural variant of the discrete logarithm problem. Therefore, we expect that $(\{-3\} \cup \mathsf{BIPSW}(n))$-Para-AI is virtually as hard as $(\{-3\} \cup \mathsf{BIPSW}(n))$-Vect-AI. While there is currently no proven reduction from Vect-AI to Para-AI, there is also no better algorithm for solving Para-AI than solving the corresponding Vect-AI problem.

This motivates us to lower bound the hardness of the $(\{-3\} \cup \mathsf{BIPSW}(n))$-Para-AI problem by the hardness of the S-Vect-AI problem, for S of same cardinality as $\{-3\} \cup \mathsf{BIPSW}(n)$. In particular, for our concrete implementation and parameters, we use the complexity analysis of the S-Vect-AI problem to roughly estimate the concrete security of our construction.

5 Putting Everything Together: PCFs for OT from Isogenies

We now recall the CPRF-to-PCF compiler of [19].

Lemma 3 (PCFs for OT from Pseudorandomly-Constrained PRFs, [19]). *Let $F\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ be a (weak) PRF. Let $\overline{F}\colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ denote the function defined by $\overline{F}_k(\cdot) := 1 - F_k(\cdot)$. Let $\mathsf{CPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ be an adaptively-secure constrained PRF supporting as constraints $\{F_k(\cdot)\colon k \in \mathcal{K}_\lambda\} \cup \{\overline{F}_k(\cdot)\colon k \in \mathcal{K}_\lambda\}$. The following construction is a (weak) PCF for oblivious transfer correlations:*

$\underline{\mathsf{PCF.KeyGen}(1^\lambda)\text{:}}$

1. *Run* $(\mathsf{pp}_0, \mathsf{msk}_0) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
2. *Run* $(\mathsf{pp}_1, \mathsf{msk}_1) \xleftarrow{\$} \mathsf{KeyGen}(1^\lambda)$.
3. *Set and output* $k_0 \leftarrow (\mathsf{msk}_0, \mathsf{msk}_1)$.
4. *Sample* $k \xleftarrow{\$} \mathcal{K}_\lambda$.
5. $\mathsf{ck}_0 \leftarrow \mathsf{Constrain}(\mathsf{msk}_0, F_k)$.
6. $\mathsf{ck}_1 \leftarrow \mathsf{Constrain}(\mathsf{msk}_1, \overline{F}_k)$.
7. *Output* $k_1 \leftarrow (k, \mathsf{ck}_0, \mathsf{ck}_1)$.
8. *Output* $(\mathsf{pp}_0, \mathsf{pp}_1)$.

$\underline{\mathsf{PCF.Eval}(1^\lambda, 0, k_0, x)\text{:}}$

1. *Parse* $k_0 = (\mathsf{msk}_0, \mathsf{msk}_1)$.
2. *Set* $r_0 \leftarrow \mathsf{Eval}(\mathsf{msk}_0, x)$.
3. *Set* $r_1 \leftarrow \mathsf{Eval}(\mathsf{msk}_1, x)$.
4. *Set and output* (r_0, r_1).

$\underline{\mathsf{PCF.Eval}(1^\lambda, 1, k_1, x)\text{:}}$

1. *Parse* $k_1 = (k, \mathsf{ck}_0, \mathsf{ck}_1)$.
2. *Compute* $b \leftarrow F_k(x)$.
3. *Compute* $r \leftarrow \mathsf{CEval}(\mathsf{pp}_b, \mathsf{ck}_b, x)$.
4. *Set and output* (b, r).

Note that while the compiler above outputs a weak PCF if it is run on a CPRF and a weak PRF, the resulting weak PCF can be upgraded to a strong PCF in the random oracle model.

Corollary 3. *There exists a PCF (as per Definition 7) for oblivious transfer correlations, under the following assumptions*

- *A random oracle H exists,*
- *A (weak) pseudorandom IPM predicate F exists; let (n, S) denote the parameters of this IPM predicate. And,*
- *(n, S)-power parallelization problem (Definition 14) is hard.*

Proof. The proof follows by combining Corollary 2 and Lemma 3. □

5.1 Achieving NIST Level 1 security.

The National Institute of Standards and Technology (NIST) has defined various security levels in its call to post-quantum cryptography standardization [53]. Following its terminology, we refer to the first NIST level of security (NIST Level 1) that characterises cryptosystems that are at least as hard to break as AES-128. This corresponds to requiring approximately 2^{128} operations to break the system by a classical adversary and 2^{64} for a quantum adversary.

In order to achieve this level of security for our CPRF, analysis of [23] and [46] suggest a key length of $n = 770$. This implies that the cardinalities of the corresponding set $S = \mathsf{BIPSW}(n)$ and $\{-3\} \cup S$ are smaller than 2^9. By the above analysis, to guarantee that our CPRF has NIST Level 1 security, we need the VECTORIZATION problem with 2^9 auxiliary inputs to require at least 2^{128} (resp. 2^{64}) operations on a classical (resp. quantum) computer to be solved. By carefully choosing the prime p that parametrizes our qt-PEGASIS EGA, see Section 4.3, according to the recent complexity analysis of the VECT-AI problem, one can obtain the desired complexity. Let us now argue that taking p such that $\log p = 1008$, which corresponds to qt-PEGASIS of level 1000 [31], is sufficient to reach this security level.

As mentioned in [39], there are currently only two articles [48,39] that study the complexity of VECT-AI while really leveraging the knowledge of the extra inputs. Both are mostly performed on the CSIDH framework, we heuristically extend the results of their analysis to our case.

The former provides a quantum analysis which suggests that, according to [39, Table 6], the CEGATs offered by isogeny-based cryptography with $\log p \approx 1000$ should guarantee at least 2^{63} bits of security against quantum adversaries. It is probable that an extensive study on this specific case would provide a larger lower bound. Firstly, the 2^{63} bound is actually stated for sets S of size 2^{12}. Hence, by reducing the size of the sets to 2^9, we may improve this lower bound 2^{63}. Secondly, one can notice that we are considering a problem which is more general than in [39]. Indeed, we do not ask our sets S to be *(super)exceptional*, i.e. such that the pairwise differences (and sums) of the elements in S are invertible

modulo the order of the acting group. This set property does not seem to serve in the complexity study conducted by the authors. However, in order to easily build such (super)exceptional sets, they are considering sets of successive integers. This is not the case in our work, and the gaps in our sets may, as mentioned in [39, Countermeasures], in fact improve the hardness of the VECT-AI problem.

In the second paper [48], it is proven that, in certain situations, the classical complexity of the VECT-AI problem is $\tilde{O}(p^{1/8})$. Which is significantly smaller than the time complexity $\tilde{O}(p^{1/4})$ of the standard VECTORIZATION problem. However, this is only applicable when the exponents of the set S are successive integers such that the largest one divides the order of the group. Hence, our construction is not affected by this attack and the classical complexity can be estimated using the standard bound $p^{1/4} \approx 2^{252}$. As a matter of fact, even if it was possible to adapt the attack, we can notice that $p^{1/8} \approx 2^{126}$ remains a large enough lower bound to consider the scheme to be secure against classical computers. In particular, the hidden logarithmic factors, for instance from the cost of $\mathbb{F}_p$ arithmetic, amply compensate for the gap with NIST level 1 requirements.

In conclusion, it is reasonable to consider that our CPRF, instantiated with qt-PEGASIS and $\log p = 1008$, has NIST Level 1 security.

5.2 Efficiency Analysis

We provide estimations on the time and memory complexity of Construction 1.

In both cases, the analyses are done for generic group actions and for the qt-PEGASIS instantiation with parameters that are believed to ensure a security level of 128 bits (see Section 5.1).

Let us begin with the memory complexity, for which generic qt-PEGASIS instantiation is also studied. The computation is done as follows: we count the number of bit strings, set elements and group elements that are part of the keys, then we estimate their bit size in a qt-PEGASIS framework. To this end, we use that, for a given $\log p$, storing a set element for the qt-PEGASIS($\log p$) framework costs $\log p$ bits, see Section 4.3, and that storing a group element costs around $\frac{3}{4} \log p$ bits, using [37].

	$\lvert\mathsf{msk}\rvert$ in bits	$\lvert\mathsf{ck}\rvert$ in bits
Generic group actions	$1 \cdot \lvert t\rvert + n \cdot \lvert g\rvert$	$\#S \cdot \lvert t\rvert + n \cdot \lvert g\rvert + n$
qt-PEGASIS($\log p$)	$(1 + \frac{3n}{4}) \log p$	$n + (\#S + \frac{3n}{4}) \log p$
qt-PEGASIS(1008), $n = 770$, $S = \mathsf{BIPSW}(n)$	583128	972986

Fig. 2. Estimating the memory cost of Construction 1 for three different instantiations: generic groups (where $|t|$ and $|g|$ respectively denote the bit size of a set element and a group element), generic qt-PEGASIS and the qt-PEGASIS instantiation achieving the security level 128 (with $\log p = 1008$, $n = 770$, and $S = \mathsf{BIPSW}(n)$, i.e., $\#S = 387$).

Extending this complexity analysis to the obtained PCF, we obtain the following key sizes for a security level of 128 bits: for the sender, the key size is $2 \cdot |\mathsf{msk}| \approx 146$ kB, while, for the receiver, it is $2 \cdot |\mathsf{ck}| - n - \#S \cdot \log p \approx 194$ kB.

Memory Optimizations for the PCF. We note that some memory optimizations used by [19] can also be applied to our construction to reduce the key sizes even more. We list them below and explain how they affect the key sizes:

- Key compression via PRGs: the elements g_i's of the master secret key can be generated by applying a pseudorandom generator on a λ-bit seed. We recall this optimization in Construction 1. This results in having a sender key of size only $2 \cdot \lambda$ bits and reduces the receiver key size by 25%.
- Exploiting the anti-periodic structure of S for BIPSW: as observed in [19], the set S defined by the BIPSW wPRF has an *anti-periodic* structure, meaning that for (almost) any $s \in [n] \setminus S$, it holds that $s - 6 \in S$ and vice versa, with exceptions only at the extremities. Using this structure, the sender of the PCF can hold only one CPRF master secret key and an additional shifting element, i.e., $k_0 = (t, g_1, \ldots, g_n, g_{n+1})$. Similarly, the receiver key can have one constrained key and a shifting element, i.e., $k_1 = ((t_s)_{s \in S}, (r^{-k_i} \cdot g_i)_{i \in [n]}, r^6 \cdot g_{n+1}, \boldsymbol{k})$. Consequently, on any input $\boldsymbol{x}$, the sender computes $y_b := \mathsf{Eval}(\mathsf{msk}, \boldsymbol{x} \| b)$, for $b = 0, 1$. The receiver also outputs (b, r), where $b = F_{\boldsymbol{k}}(\boldsymbol{x})$ and $r = \mathsf{CEval}(\mathsf{ck}_{(S,\boldsymbol{k})}, \boldsymbol{x} \| b)$. The correctness will be maintained because of the following: if $b = 0$, then $\langle \boldsymbol{x}, \boldsymbol{k} \rangle \in S$, and otherwise $\langle \boldsymbol{x}, \boldsymbol{k} \rangle - 6 \in S$. This technique significantly reduces the receiver's key size, almost by half.

Combining the two techniques above, we can get a key size of $\lambda = 128$ bits for the sender and an expected value of $n + \#S \cdot \log p + 3/4 \cdot 3/4(n+1) \log p = 828023$ bits, or ≈ 103 kB, for the receiver.

To analyze the time complexity, we start by enumerating the operations done in each algorithm, namely sampling and multiplying group elements and evaluating group actions. From that, we compute an estimated runtime for a low-level implementation of the scheme using the qt-`PEGASIS` framework with 128-bit security level. To this end, we rely on different sources to obtain a reasonable runtime for each operation.

The recent C implementation given in [31] allows us to evaluate qt-`PEGASIS`(1008) group actions in approximately 146ms. On the other hand, we bound the runtime for sampling a random ideal by 2ms, following the Rust implementation of [22]. Finally, we consider composition of ideals to be negligible compared to the rest of the algorithm.

We emphasize that this method does not provide a precise runtime, as it relies on different low-level implementation results which are using different languages and configurations; however, we believe it gives a meaningful estimation of a time complexity that is achievable in practice. In addition, the proof of concept provided with this work is not intended to accurately estimate the running time of Construction 1 in a low-level implementation. In particular,

sampling and multiplying ideals are not optimized procedures. While their cost should be negligible compared to the group action evaluations, the current implementation spends more than 50% of the KeyGen running time performing ideal manipulations.

	KeyGen	Eval	Constrain	CEval
Generic group actions	$1 \cdot A + (n+1) \cdot I$	$1 \cdot A + n \cdot M$	$\#S \cdot A + 1 \cdot I + n \cdot M$	$1 \cdot A + n \cdot M$
qt-PEGASIS(1008), $n = 770$, $S = \mathsf{BIPSW}(n)$	$\simeq$ 1688ms	$\simeq$ 146ms	$\simeq$ 56504ms	$\simeq$ 146ms

Fig. 3. Estimations of the time cost of Construction 1 for two different instantiations: generic groups (where I denotes the ideal sampling cost, A denotes the group action evaluation cost and M denotes the ideal mulitplication cost) and the qt-PEGASIS instantiation achieving security level 128 (with $\log p = 1008$, $n = 770$ and $S = \mathsf{BIPSW}(n)$).

The numbers translate to $1000/146 \approx 7$ OTs/sec for the resulting PCF.

5.3 Key Compression of the CPRF

We show how to compress the keys using PRGs, inspired by [19]. The construction is defined over binary inputs and constraints, i.e., $\boldsymbol{x}, \boldsymbol{z} \in \{0,1\}^n$.

Construction 1: No-Evaluation Unpredictable CPRF for IPM

- Let $n \in \mathsf{poly}(\lambda)$, $S \subset [n]$, and $(G, T, \star)$ be an AEGA.
- Let τ denote the bit-size of an element in T. Let $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{(\tau+n)\lambda}$ and $\mathsf{G}' : \{0,1\}^\lambda \to G$ be pseudorandom generators.

$\mathsf{KeyGen}(1^\lambda)$:

– Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$.
– Output $\mathsf{msk} := (\mathsf{seed})$.

$\mathsf{Eval}(\mathsf{msk}, \boldsymbol{x})$:

– Compute $(t, \mathsf{seed}_1, \ldots, \mathsf{seed}_n) \leftarrow \mathsf{G}(\mathsf{seed})$.
– For $i \in [n]$, compute $g_i \leftarrow \mathsf{G}'(\mathsf{seed}_i)$.
– Output $(g_1^{x_1} \cdot \cdots \cdot g_n^{x_n}) \star t$.

$\mathsf{Constrain}(\mathsf{msk}, (S, \boldsymbol{k}))$:

– Parse $\mathsf{msk} = (\mathsf{seed})$.
– Compute $(t, \mathsf{seed}_1, \ldots, \mathsf{seed}_n) \leftarrow \mathsf{G}(\mathsf{seed})$.

– Sample $r \xleftarrow{\$} G$.
– $\forall s \in S$, define $t_s := r^s \star t$.
– $\forall i \in [n]$:
  • If $k_i = 0$, set $\widetilde{\alpha}_i = \mathsf{seed}_i$.
  • If $k_i = 1$, set $\widetilde{\alpha}_i := r^{-k_i} \cdot g_i$.
– Let $\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_1, \ldots, \widetilde{\alpha}_n)$.
– Output $\mathsf{ck}_{(S,\boldsymbol{k})} := ((t_s)_{s\in S}, \widetilde{\boldsymbol{\alpha}}, \boldsymbol{k})$.

$\mathsf{CEval}(\mathsf{ck}_{(S,\boldsymbol{k})}, \boldsymbol{x})$:

– Parse $\mathsf{ck}_{(S,\boldsymbol{k})} = ((t_s)_{s\in S}, \widetilde{\boldsymbol{\alpha}}, \boldsymbol{k})$.
– For $i \in [n]$:
  • If $k_i = 0$, set $\alpha_i = \mathsf{G}'(\widetilde{\alpha}_i)$.
  • If $k_i = 1$, set $\alpha_i := \widetilde{\alpha}_i$.
– Let $s_{\boldsymbol{x}} = \langle \boldsymbol{x}, \boldsymbol{k} \rangle$. If $s_{\boldsymbol{x}} \notin S$, output $\perp$.
– Otherwise, output $(\alpha_1^{x_1} \cdot \cdots \cdot \alpha_n^{x_n}) \star t_{s_{\boldsymbol{x}}}$.

5.4 Implementation

We implement our CPRF in SageMath/Python for the BIPSW inner-product membership predicate at https://github.com/IRIFCrypto/cprf-from-isogenies.

Acknowledgements. This work is supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement ISOCRYPT – No. 101020788), by the Research Council KU Leuven grant C14/24/099, by CyberSecurity Research Flanders with reference number VOEWICS02, by the HQI initiative (ANR-22-PNCQ-0002), and by the PEPR quantique France 2030 programme (ANR-22-PETQ-0008).

References

1. Michel Abdalla, Thorsten Eisenhofer, Eike Kiltz, Sabrina Kunzweiler, and Doreen Riepel. Password-authenticated key exchange from group actions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 699–728. Springer, Cham, August 2022.
2. Shweta Agrawal, Kaartik Bhushan, Geoffroy Couteau, and Mahshid Riahinia. Post-quantum public-key pseudorandom correlation functions for OT. Cryptology ePrint Archive, Paper 2026/877, 2026.
3. Navid Alamati, Luca De Feo, Hart Montgomery, and Sikhar Patranabis. Cryptographic group actions and applications. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 411–439. Springer, Cham, December 2020.

4. Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Cham, August 2019.
5. Benny Applebaum and Pavel Raykov. Fast pseudorandom functions based on expander graphs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part I*, volume 9985 of *LNCS*, pages 27–56. Springer, Berlin, Heidelberg, October / November 2016.
6. Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for NC^1 in traditional groups. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 543–574. Springer, Cham, August 2018.
7. Karim Baghery, Daniele Cozzo, and Robi Pedersen. An isogeny-based id protocol using structured public keys. In Maura B. Paterson, editor, *Cryptography and Coding*, pages 179–197, Cham, 2021. Springer International Publishing.
8. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012.
9. Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 97–109. Springer, Berlin, Heidelberg, August 1995.
10. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th ACM STOC*, pages 479–488. ACM Press, May 1996.
11. Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 227–247. Springer, Cham, December 2019.
12. Jean-François Biasse and Fang Song. Efficient quantum algorithms for computing class groups and solving the principal ideal problem in arbitrary degree number fields. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 893–902. SIAM, 2016.
13. Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Cham, November 2018.
14. Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 520–550. Springer, Cham, December 2020.
15. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Cham, August 2022.
16. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *61st FOCS*, pages 1069–1080. IEEE Computer Society Press, November 2020.
17. Gilles Brassard and Moti Yung. One-way group actions. In Alfred J. Menezes and Scott A. Vanstone, editors, *CRYPTO'90*, volume 537 of *LNCS*, pages 94–107. Springer, Berlin, Heidelberg, August 1991.

18. Lennart Braun, Geoffroy Couteau, Kelsey Melissaris, Mahshid Riahinia, and Elahe Sadeghi. Fast pseudorandom correlation functions from sparse LPN. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part VII*, volume 16251 of *LNCS*, pages 439–473. Springer, Singapore, December 2025.
19. Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part VI*, volume 14656 of *LNCS*, pages 88–118. Springer, Cham, May 2024.
20. Wouter Castryck and Thomas Decru. CSIDH on the surface. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 111–129. Springer, Cham, 2020.
21. Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 395–427. Springer, Cham, December 2018.
22. Kostas Kryptos Chalkias, Jonas Lindstrøm, and Arnab Roy. An efficient hash function for imaginary class groups. Cryptology ePrint Archive, Report 2024/295, 2024.
23. Jung Hee Cheon, Wonhee Cho, Jeong Han Kim, and Jiseung Kim. Adventures in crypto dark matter: Attacks and fixes for weak pseudorandom functions. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 739–760. Springer, Cham, May 2021.
24. H. Cohen and H. W. Lenstra. Heuristics on class groups of number fields. In Hendrik Jager, editor, *Number Theory Noordwijkerhout 1983*, pages 33–62, Berlin, Heidelberg, 1984. Springer Berlin Heidelberg.
25. Leonardo Colò and David Kohel. Orienting supersingular isogeny graphs. *Journal of Mathematical Cryptology*, 14(1):414–437, 2020.
26. Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. QuietOT: Lightweight oblivious transfer with a public-key setup. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 197–231. Springer, Singapore, December 2024.
27. Geoffroy Couteau and Clément Ducros. Pseudorandom correlation functions from variable-density LPN, revisited. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part II*, volume 13941 of *LNCS*, pages 221–250. Springer, Cham, May 2023.
28. Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Cham.
29. Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291, 2006.
30. Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The TinyTable protocol for 2-party secure computation, or: Gate-scrambling revisited. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 167–187. Springer, Cham, August 2017.
31. Pierrick Dartois and Max Duparc. Chasing rabbits through hypercubes: Better algorithms for higher dimensional 2-isogeny computations. Cryptology ePrint Archive, Paper 2026/114, 2026.
32. Pierrick Dartois, Jonathan Komada Eriksen, Tako Boris Fouotsa, Arthur Herlédan Le Merdy, Riccardo Invernizzi, Damien Robert, Ryan Rueger, Frederik Vercauteren, and Benjamin Wesolowski. PEGASIS: Practical effective class group

action using 4-dimensional isogenies. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part I*, volume 16000 of *LNCS*, pages 67–99. Springer, Cham, August 2025.
33. Pierrick Dartois, Jonathan Komada Eriksen, Riccardo Invernizzi, and Frederik Vercauteren. sfqt-sfPegasis: Simpler and faster effective class group actions. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part IV*, volume 16545 of *LNCS*, pages 636–665. Springer, Cham, May 2026.
34. Luca De Feo, Tako Boris Fouotsa, Péter Kutas, Antonin Leroux, Simon-Philipp Merz, Lorenz Panny, and Benjamin Wesolowski. SCALLOP: Scaling the CSI-FiSh. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 345–375. Springer, Cham, May 2023.
35. Christina Delfs and Steven D. Galbraith. Computing isogenies between supersingular elliptic curves over $\mathbb{F}_p$. *DCC*, 78(2):425–440, 2016.
36. Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Authenticated garbling from simple correlations. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part IV*, volume 13510 of *LNCS*, pages 57–87. Springer, Cham, August 2022.
37. Samuel Dobson, Steven Galbraith, and Benjamin Smith. Trustless unknown-order groups. *Transactions on Mathematical Cryptology*, 1(2):25–39, Mar. 2022.
38. Julien Duman, Dominik Hartmann, Eike Kiltz, Sabrina Kunzweiler, Jonas Lehmann, and Doreen Riepel. Generic models for group actions. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 406–435. Springer, Cham, May 2023.
39. Paul Frixons, Valerie Gilchrist, Péter Kutas, Simon-Philipp Merz, Christophe Petit, and Lam L. Pham. Another look at the quantum security of the vectorization problem with shifted inputs. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part I*, volume 16541 of *LNCS*, pages 633–662. Springer, Cham, May 2026.
40. S Galbraith, L Panny, B Smith, and F Vercauteren. Quantum equivalence of the dlp and cdhp for group actions math. *Cryptol*, 1(1):40, 2021.
41. Oded Goldreich. Candidate one-way functions based on expander graphs. Cryptology ePrint Archive, Report 2000/063, 2000.
42. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.
43. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
44. Sebastian Hasler, Pascal Reisert, and Ralf Küsters. Pseudorandom correlation functions from ring-LWR. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part VII*, volume 16251 of *LNCS*, pages 401–438. Springer, Singapore, December 2025.
45. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Berlin, Heidelberg, August 2003.
46. Thomas Johansson, Willi Meier, and Vu Nguyen. Differential cryptanalysis of mod-2/mod-3 constructions of binary weak prfs. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 477–482. IEEE, 2023.
47. Taechan Kim. Multiple discrete logarithm problems with auxiliary inputs. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 174–188. Springer, Berlin, Heidelberg, November / December 2015.

48. Taechan Kim. Security analysis of group action inverse problem with auxiliary inputs with application to csidh parameters. In *International Conference on Information Security and Cryptology*, pages 165–174. Springer, 2019.
49. Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. In *36th FOCS*, pages 170–181. IEEE Computer Society Press, October 1995.
50. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.
51. Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Berlin, Heidelberg, August 2012.
52. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.
53. NIST. Post-quantum cryptography standardization. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography, 2016.
54. Hiroshi Onuki. On oriented supersingular elliptic curves. *Finite Fields and Their Applications*, 69:101777, 2021.
55. Lorenz Panny, Christophe Petit, and Miha Stopar. KLaPoTi: An asymptotically efficient isogeny group action from 2-dimensional isogenies. *CiC*, 2(3):5, 2025.
56. Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 602–632. Springer, Cham, August 2023.
57. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Cham, August 2022.
58. Sacha Servan-Schreiber. Constrained pseudorandom functions for inner-product predicates from weaker assumptions. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part II*, volume 15485 of *LNCS*, pages 232–265. Springer, Singapore, December 2024.
59. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.
60. Joseph H. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer New York, New York, NY, 2009.
61. William C. Waterhouse. Abelian varieties over finite fields. *Annales scientifiques de l'École Normale Supérieure*, Ser. 4, 2(4):521–560, 1969.